第1章 软件测试行业概貌

读者朋友，你好！虽然你我未曾谋面，但很感谢你能翻开这本书。本书的书名叫《软件测试工程师面试指导》，但我们先别急着谈论面试的事情，不管你有没有明确的想法要加入到软件测试这个新兴的行业中来，先了解一下这个行业的基本情况都是一种合适的做法。

1.1 什么是软件测试

因为IT行业比任何传统行业都要年轻得多，而软件测试行业更是“幼稚”得不得了，所以我们将通过分析一个传统行业来对软件测试行业进行说明。大家在日常生活中购买一些产品的时候，常常能在它的包装里发现一张质量检验卡，电器产品尤其是这样。质量检验卡上有质检员的姓名或者编号，表明这件产品经过了检验，可以放心使用。虽然我们并不确定是否真的有人做了检验☺，但这是一套很好的机制。偶有耳闻从啤酒里喝到异物的怪事，此类事情的发生，也有质检员的责任。软件测试工程师就相当于质检员的角色，在软件研发的周期中验证当前的软件产品是否是客户想要的产品。软件测试就是要发现问题，及时提出，推动开发人员及时修正，最终的目的是让公司能够按时保质地向客户交付软件产品。

与任何其他产品相比，软件产品很特殊。它不是由钢筋混凝土堆砌而成，而是由一行一行的代码构成。因为代码的不可见性，所以有的朋友（包括公司老板和客户）就会有些误解：软件出了错怕什么？改改不就行了？反正不用拆墙凿洞的。因此，往往我们对建筑的质量抓得很紧，每一车的混凝土都要取样，而对软件产品的质量却会听之任之。认为修改软件错误很容易，这是一个很大的误解。举一个例子，我们不说Windows操作系统这个庞大的航空母舰，就以Word为例，如果你发现它不能稳定工作，是很难查错的。要想从千万行的代码中找到出错原因并修正它，和在原始森林里捉迷藏没有什么区别。为了避免这种被动，软件测试就是要在软件研发的过程中，尽可能及时发现问题。而这，需要软件测试工程师的辛勤劳动。

大家都知道，公司里的任何一个员工都是要拿薪水的，为什么需要单独做软件测试的人呢？让开发工程师自己做测试不就行了吗？☺，实际情况不是这样的。任何父母，打自己孩子的时候都会三思而行，能不打就不打；任何人，要向大家公开自己的不足的时候，都会想到掩盖或者避重就轻，

能不说就不说。一行行代码构成的软件产品对于开发人员来说，就好像自己的孩子，要从自己孩子身上找缺点进而进行处罚，是一件难事，效率也会很差。这种事情，的确需要另一种角色来做，需要一个唱黑脸的人。这是心理上的因素。同时，从工作量上来说，测试的工作复杂而重复，开发人员不可能在开发之余还能做好测试。

如果大家了解海尔的故事，可能对张瑞敏砸冰箱的故事有一点印象。海尔创办之初，产品质量上不去，冰箱卖出去了，又被退回一部分。张瑞敏很有魄力，把职工都叫到一块，当众把退回的冰箱砸了，以此让大家树立质量意识。我们知道，软件产品发布的时候是刻成光盘的，所有光盘里的内容是完全一样的，也就是说，真到了软件公司“砸冰箱”的时候，要砸的是所有的“冰箱”，而不只是一部分，这种结果恐怕是很多公司无力承受的。所以，软件测试很重要，是吧？

软件产品在社会发展中起着越来越重要的作用，软件测试也肩负着越来越重要的责任。我们无法想象，如果医疗软件出现问题会怎样？例如手术中，医生得到了错误的心电图；我们也无法想象，如果宇宙飞船的操作系统死机怎么办？我自己曾亲身经历过一次软件事故，因为一个bug致使收费系统不能工作，导致高速公路不能通行，高速公路出入口堵得一塌糊涂，电视台都对其进行了报道，教训很惨痛。如果你准备加入这一行，一定要有责任心。

虽然现在找工作的行情并不乐观，但是我们可以分析一下大的趋势，打一个形象的比喻，我们中国就像一架飞机，刚刚通过快速滑行已进入爬升的阶段。国家的快速发展催生了很多行业对信息化的需求，而每一个信息化的需求中都包含有大量的对软件测试工程师的需求，例如银行系统、医院系统、工商管理系统、税务管理系统、水利管理系统、证券系统、校园管理系统，等等。比较一下你的家乡和读书（或者工作）所在的城市的信息化程度的差别，就可以知道市场有多广阔，而本身我们国家也正在努

力缩小与发达国家在信息化上的差距。而且，经过这些年的发展，软件测试也逐步得到了软件行业的认可，现在老板们也意识到，要开发一个软件，不光需要软件开发工程师，测试工程师也是不可或缺的。所以，现在整体氛围很好，是加入软件测试行业的好时候。

1.2 软件测试行业的现状和就业前景

要了解软件测试行业，如果有一份专业的调查报告毫无疑问是最好的。我在网上看到一份软件行业的调查报告（正式出版物）竟然能卖到几千元的高价，让我等平头百姓不禁又捏了捏兜里仅剩的几张钞票，暗自感叹道："难道没有别的办法了吗？"。所以我想，我们可以尝试以我们自己的角度来看这个行业，其中再引用一些数据，不一定可靠的感觉和确凿的数据混在一起，得出我们自己的结论。

要了解软件测试行业，最好还是从软件行业谈起，毕竟软件测试行业只是软件行业的一个分支。软件行业的整体情况就是软件测试行业的大环境，只有软件行业蓬勃发展，才会有软件测试业的如火如荼。

你认为这些年或者就说近几年，软件行业有没有发展呢？我认为是有很大发展的。我先掰着手指头说说我的理由：

（1）国际软件巨头的名字在国内已经是耳熟能详了。他们在中国设立公司和研发机构，客观上会推动中国软件业的发展，至少增加了就业机会。

（2）逐渐出现了一些知名的国内软件企业。虽然他们无法和国际软件巨头相比，但羽翼也逐渐丰满，在国内市场有着不俗的表现。

（3）虽然IT还是被人称为"挨踢"，但是相对来说这个人群还是一个高收入的人群。

（4）软件外包越来越多地被大家提起，也有了越来越多的软件外包公司。身边有了更多的朋友在做外包，也有更多的人亦真亦假地关注中国和

印度的外包业务发展的差距以及其中的原因。在国内出现了几千人的外包公司，虽然公司大并不表示实力一定强，但规模大对于普通劳动者来说却是福音，工作机会多啊。

（5）软件的价值逐渐得到体现。软件只是合同中被赠与的一项的情况越来越少了，客户方愿意为软件买单了。

（6）国家打击盗版力度加大了，例如北京中关村卖盗版盘的人少了很多，在这方面老盯着我们的美国也承认中国有了进步。盗版少了，商业产品软件才有可能存活。

下面是一组数据：

2006年第四季度国内ERP（Enterprise Resource Planning，企业资源规划和管理）市场：

销售额（亿元）	比上季度增长	比去年同期增长
13.15	36.67%	20.40%

（数据来源：易观国际《2006年第四季度中国ERP市场数据监测》）

ERP是软件行业里很有活力的一个分支，从上表中我们可以看到它的增长强劲。

软件行业整体收入：

年度	软件行业销售收入（亿元）
2001	796
2006	4800

（数据来源：http://www.csiaexpo.com/softnews/htm/focus/view/1552-07@ 2007-01-30.html）

从上表中我们可以看到，这5年间软件行业整体收入的增长是惊人的，几乎是每年的增长额都相当于2001年的总收入，真是不简单。所以说，软件行业是在快速地发展。

我们再来看软件测试这个行业，你觉得这几年软件测试行业的情况怎样呢？我从事软件测试工作有5年多了，我想用一个词来形容：突飞猛进。

（1）软件测试的职位多了。无论上哪个招聘网站，都能搜索出很多软件测试工程师和测试管理的职位。

（2）软件测试方面的书多了。

（3）做软件测试的朋友多了。

（4）出现了软件测试方面的专业网站。

（5）“软件测试”这个词的曝光率越来越高，软件测试甚至被称为“黄金职业”或者“职业新宠”，有很多关于软件测试工程师供不应求的报道。当然，也不排除这里面可能有一点炒作和起哄的成分☺。

（6）软件测试工程师的待遇提高了。就拿外包行业来说，测试工程师不一定比开发工程师赚得少。

（7）软件测试的吸引力增强了，有越来越多的人愿意加入到这个行业中。

（8）出现了多家专门的软件测试培训机构。他们的商业模式是为想要成为软件测试工程师的朋友提供培训服务，如果软件测试行业没有大的发展，他们怎么可能存活下来？

我们再来看组数据：

企业对软件测试的认同	比例
非常重要	68.2%
比较重要	31.8%
只起到一定作用	0%
可有可无	0%

（数据来源：智联招聘《2006年度软件测试行业专项调查报告》）

上表显示了企业对软件测试的认同的提高，表示老板们愿意为软件测试花钱，这是个好的信号。

测试人员 ： 开发人员	比例
1 : 5	36.4%
1 : 2 以及以上	31.8%
1 : 1以及以上	31.7%

（数据来源：智联招聘 《2006年度软件测试行业专项调查报告》）

上表所显示的比率已经很不错了，要知道，在2000年，我所在的一家中等规模的公司，研发部里没有一位专职的测试工程师。虽然测试人员与开发人员的比率没有一个统一的约定，不同的软件公司都有自己的实际情况，但是测试人员比率过小的确不利于软件测试的进行，进而影响到软件质量。随着软件测试逐步得到重视，软件公司会增大对软件测试的投入，留出更多的测试职位，让测试工程师足以与开发工程师相抗衡。

综上所述，我想我们可以下一个结论：软件测试这个行业正在蓬勃发展。

1.3 谁适合软件测试这个行业

虽然前面我们得出结论：软件测试行业“形势一片大好”，但在考虑个人职业发展道路的时候，我们始终都要落实到自己的实际情况上来。软件测试行业发展很好、前景不错，这当然很好，但是否适合自己呢？究竟谁适合软件测试这个行业呢？这种宏观和微观相结合的思考问题的方法，才是一种负责任的做法。

在软件测试行业有很多职位，例如软件测试工程师、高级软件测试工程师、测试Leader（组长，带领一个小团队）、测试经理、测试总监，等等。我们这里以软件测试工程师为例，来讨论谁适合做软件测试工程师，其他

的职位是后续的事情，现在不必管它们。

做软件测试工程师的一个基本要求是宽泛而扎实的计算机基础知识。不同的软件产品，采用的技术各不相同，测试工程师需要有宽泛的计算机基础知识才能满足它们的要求，例如操作系统、数据库、数据结构、编程语言、测试理论，等等。

耐心。客观上来说，做什么事情都要有耐心。工作没做完的时候你就是不能休假，老板批你的时候你就得听着，这些在耐心方面的要求是普遍的要求，而软件测试的要求更高。一个软件测试用例，在整个软件研发周期内，你可能要执行它几遍，十几遍；一个bug，可能要在开发和测试人员之间来回数次。测试工作的重复性很高，没有耐心怕是如同嚼蜡，索然无味。

细心。要发现软件里的毛病，不经过严密的逻辑思维，不经过细心的排查，是不行的，大白天拣金元宝的事情怕是很少。

同时，有一点我也想让你知道，学历不是很重要。在计算机相关行业，只要你有真才实学，用人单位降低学历门槛的可能性是很大的。我有个朋友中专毕业做软件测试，做得很好。

非计算机专业的朋友也有机会做软件测试。如果你是计算机相关专业的当然更好，而如果不是，也还是有机会的。我们国内的软件业是以行业软件取胜的，例如财务软件、税务软件、水利管理软件、银行软件等，软件公司在招聘测试工程师的时候，除了考虑计算机专业背景的学生以外，相关专业的学生也在考虑范围之中。例如，如果你是财务专业的，可以到做财务软件的公司去试试，你的专业知识是你的亮点。

客观上来说，软件测试的门槛的确要低一些，有利于在工作经验方面没有优势的朋友介入进来。从长远上来看，软件测试也有利于一个人的经验积累，经验越多越好，也就是“越老越值钱”。但是，做一个优秀的测试工程师和成为一个优秀的开发工程师同样不容易，你在决心进入这个大

门的时候就要记住这一点，不要掉以轻心啊。

欢迎你迈进软件测试这个行业的大门，并祝你好运！

1.4 培训和认证

1. 软件评测师

软件评测师是国家软件水平考试的一部分，由信息产业部和人事部主管，具体由信息产业部电子教育与考试中心来实施。一年有两次考试机会，但也是轮考制，也就是说并不一定每次都会有软件评测师的考试，具体信息可以查询http://www.ceiaec.org。

从互联网上可以轻松获取软件评测师的考试大纲，在书店也可以买到考试大纲和指定教材。软件评测师考试的考核重点有：

（1）考生应当具备宽泛的计算机基础知识，包括操作系统、数据库、中间件、程序设计、网络等；

（2）考生应当熟悉软件工程知识；

（3）考生应当熟悉软件测试方法和标准；

（4）考生应当掌握C/C++或者Java语言；

（5）英文良好。

这个考试还有一个很好的地方就是与职称挂钩。软件评测师是中级考试，如果通过可以获得工程师的职称，可谓一举两得。

考试费用不高，考试的范围也不是特别宽泛，有兴趣的朋友可以试试，详情请查询http://www.ceiaec.org。

2. Mercury 系列认证

Mercury有一系列的软件测试产品，例如LoadRunner、WinRunner、TestDirector、QuickTestPro等，这些工具在测试业界内都是很知名的。Mercury公司在销售这些软件产品之余，还设计了相关的认证，它们是：

（1）SP（Specialist Product，产品专家）；

（2）CPC（Certified Product Consultant，产品顾问认证）。

以上两种认证会和具体产品相结合，例如LoadRunner SP或者LoadRunner CPC。对于考生来说，Mercury的认证在培训和考试上都有一定的花费，具体额度请以Mercury公布的为准。就国内的就业市场来说，这个证书不是必须的，但是，有证书比没有肯定要好。

此外，中国软件评测中心也做软件测试工程师的培训和认证工作（初级、中级、高级），单独发证（与职称没有对应关系），有兴趣的朋友可以到它的网站获取相关信息。

1.5 软件测试工程师的职业前景

我现在在北京工作，北京是个很有活力的城市，也因此给人一种假相：北京好像是一座只有年轻人的城市。上班的时候你在公交车或者地铁上举目望去，全是年轻人。在公司也是这样，四十岁以上的人很少，尤其在计算机公司，少数几个可能已经是老总了。我不知道中年和老年人哪里去了。这样的环境很有活力，但也给人在心理上产生一种压力：以后人到中年怎么办？去哪里？还有没有工作？做软件测试有前途吗？

以后的发展如何，是计算机从业人员常挂在嘴边的一个话题。我们国家的职业竞争的确很残酷，但是，我想任何一个行业都不会拒绝有经验的人，到时候虽然年龄大一些，只要我们足够好，工作肯定会有的。

或许说四十岁以后的事情有点早，谁都不知道以后会怎样。我们大概可以展望一下，在现在的环境背景下，这几年测试工程师的职业发展前景如何呢？

（1）可以成为一个资深的测试工程师。如果我们喜欢，如果我们的确没有机会被提升或者是不喜欢管理岗位，那么我们可以做资深的测试工程

师。集多年的工作经验于一身，加上敬业精神，照样可以把工作做得很好，可以得到大家的认同和尊重。大家会因为一个人有头衔而尊重他，更会因为他有真才实学、丰富的经验而敬佩他。

（2）测试管理者。带领一个团队或者一个公司，去做更大的事情。在一些大的公司，测试这条线上有很多职位，像Leader、Manager，等等。如果有足够信心，也可以自己办一个软件测试公司，在市场上拼一拼。

（3）测试咨询和测试培训人员。新人需要培训，新公司实施测试方案需要有人指导，这时候我们可以把自己的经验转化为另一种生产力。

（4）测试书籍编写者或者翻译者。

前景已经展望完了，剩下的就是要我们足够努力，做到足够好。

1.6 软件测试知名网站

51Testing：www.51Testing.com

测试时代：www.testage.org

为了避免做“托儿”的嫌疑，培训机构我就不做推荐了，有需要的朋友可以从互联网上获取相关资料。

02

第2章 软件测试从业人员的基本要求

当你在软件测试这个行业的大门前徘徊的时候，我知道☺，你可能在琢磨：如果要做软件测试的话，有什么要求吗？也不知道自己是否适合？在你看过了本章之后，或许你对这个问题就会有了一个大概的认识。我鼓励你来做软件测试，只要自己觉得满足了一些基本的要求，你肯定行的☺，事在人为嘛。同时，我也鼓励你去和现在已经是测试工程师的朋友交流（经过他人介绍也可以啊），不同角度的信息会有利于你的思考。

2.1 软件测试的入门要求

大概一年半前，我在一篇文章里写道：

“有些事情说起来有些滑稽，几年前，我们说，软件测试业在中国刚刚起步。现在，我们还是在说，软件测试业还是刚刚起步。几年的时间，对于其他行业是一个什么概念我不清楚，但对于IT行业来讲，不是一个短的时间。实际上，国内软件业的发展缓慢限制了测试业的发展，没有软件的开发计划，测试的需求不旺啊。然而，反过来想，在软件测试业还没有大红大紫的时候介入进去，应该是个好时候。”

今天，我必须承认，我的这段话有点落伍了，短短一年多的时间，软件测试业已经崭露出强劲的发展势头，已经广为人知，越来越为软件行业和个人所接受。但，我要说，它还是刚刚起步，因为我们国家的软件业也还在起步阶段，现在的知识产权环境有了很大的改善，软件作为智力密集型产品得到了国家和国民的认可，所以，软件测试的春天也是刚刚开始。

那么，从事软件测试的基本要求是什么？我根据自己的体会，罗列一些，供朋友们参考。

首先，要有宽泛的计算机基础知识。微机原理、数据结构、数据库、操作系统原理、编译原理、逻辑、编程语言、网络，等等，都要系统地学习过。都精通不大可能，因为各人的兴趣都不相同，但是，这些功课的基本知识点是应当了解的。在谈到职业类别的时候，我们可以说C程序员、C#程序员、Java程序员，而没有说C测试员、C#测试员、Java测试员、程序员可以只擅长某一门编程语言，测试员却不行，为什么呢？因为测试员是代表用户的，在做测试的时候，测试员需要考虑到方方面面的事情。例如，如果测试一个用C语言编写的上网拨号程序，测试员需要考虑：

（1）程序的功能是否正确（要求计算机知识）；

（2）是否符合用户的使用习惯（要求界面设计知识和换位思考能力）；

（3）性能是否满足要求，例如长时间使用下的性能及稳定性（要求深入的计算机知识）；

（4）是否能够满足用户在不同操作系统上使用的要求（要求计算机知识）；

（5）如果在全球发布，是否满足不同语言和文化的需求（要求软件国际化测试知识）；

（6）如何搭建测试环境（要求动手能力，硬件知识）；

（7）做代码检查（要求比较深入的C语言知识）；

（8）...

从上面列出的要求来看，各方面知识都了解一点，你在做测试的过程当中就会感觉顺手得多。如果某些方面还差一些，没有关系，计算机行业的特点就是边做边学，只要是个有心人，学习是很快的。

其次，要掌握一门编程语言。有的朋友可能会说，我就是不愿意做编程才来做测试的，怎么做测试还有这个要求？我要尝试说服你☺。我的理由有两个：

（1）只有知道怎么做一个软件产品，即编码技术，才能真正懂得这个产品。而只有真正懂得了产品，才能做好测试。一行代码都不会，你会始终是个门外汉。不要只满足于点鼠标，而要去尝试着打开我们面前的黑盒子。

（2）自动化测试技术需要编程技术。自动化测试是软件测试的一个发展方向，一方面很多测试工具都需要人工干预，编写代码；另一方面在有的情况下需要自己编写测试工具。

对于测试员来说，编程技术不要求精通，但一定要会。

再次，学好英语。在现阶段，我们只能承认，在计算机方面，英语国家领先。有很多的资料都是英语的，如果仅仅局限在中文资料方面，会影响你的知识的渊博程度☺。举一个简单的例子，Windows操作系统会捕捉到

一些程序或者操作系统内部的异常，你可以根据这个异常的编号到微软网站上去查找错误原因和解决办法，其中有很大一部分资料是英文的，也许还没有翻译过来或者以后也不会翻译的。

以上所说的几点看法，都是在计算机行业里面打转，下面说几个“虚”的要求吧。

（1）锻炼出一双测试的眼睛。我的一个朋友，她也是做软件测试的，有一次她和她老公去买笔记本电脑，她一眼就看出液晶屏上有几个坏点，而她老公却看不出来。她说，这要归功于她有一双测试的眼睛。测试的眼睛，就是对问题特别敏感，能够发现常人发现不了的问题。测试员就是要找出软件中的问题，有了这双眼睛会让你收益非浅。耐心、细心和经验会有助于我们达到这个要求。

（2）平和的心态。从心理学上说，每个人都不喜欢别人对自己挑毛病，程序员也是这样。所以，要以平和的心态去看待发现的软件问题，以平和的心态去和程序员交流。千万不要以为自己发现了几个问题，就可以责怪程序员，或者冲过去骂他们一顿。也不要在背后谈论谁谁不行，bug太多。一个项目是大家共同做的，需要聚集体之力才能做完。我们测试员发现的问题多一些，表明项目的风险又少了一点，应该高兴才是。如果你的脾气不好，可能这个恶名会掩盖你的真才实学，很可惜的。

上面所说的，都是我个人的见解，如果你觉得自己还有些差距而滋生了一些犹豫，我希望没有把你已经提起来准备跨进软件测试大门的脚吓回去。你能不能做软件测试，很多时候都是要看你的工作机会，这并没有一个死框框在做限定，说谁一定能做，谁一定不能做。现在有一部分的测试工作，对计算机知识的要求并不高，例如ERP软件测试，老板可能更希望你有财务专业背景，或者人事专业背景，或者物流行业的实际工作经验。所以，即使不是计算机专业的，也还是有机会做测试的。当然，如果你想做得深入一点，还是需要再多学一点计算机知识。

好了，我罗嗦完了，你准备好了吗？让我们一起来努力吧。最重要的，把信心带上。

2.2 应当具备的计算机专业知识

有的时候，想从事软件测试工作的朋友写信问我，如果要做软件测试的话，究竟要懂哪些东西才可以呢？

诚然，我在前面说过，非计算机专业的也可以做软件测试，特别是在一些对行业知识要求高的领域，例如财务软件等。但是，我相信，每一位朋友在考虑发展方向的时候都会做长期的考虑，如果我们想在这个行业内走得更远，想适应更多的项目，一些计算机专业的基础知识是应当学习和掌握的。是不是计算机专业出身并不重要，重要的是具备这些基础知识。

在讨论具体的专业课之前，我想说明一下的是，请学好数学，数学课主要是训练我们的逻辑思维能力，它是我们编制程序和做软件测试的基础。下面我就几门重要的计算机基础课程做一个讨论，供大家参考：

1. C 语言

（1）数据类型；

（2）运算符；

（3）数组；

（4）程序控制流：

- If－else
- Else－if
- For
- While
- Do－while
- Switch
- Break和continue

- Goto

（5）函数；

（6）指针；

（7）结构；

（8）头文件。

2. C++语言

（1）面向对象的编程方法；

（2）类和对象；

（3）构造函数和析构函数；

（4）公有，私有和受保护类型；

（5）继承和派生；

（6）多态；

（7）虚函数；

（8）掌握一种可视化C++编程工具，例如VC++。

3. 操作系统

（1）操作系统的几种类型：批量、分时和实时操作系统；

（2）进程；

（3）进程同步和互斥；

（4）进程间的通信；

（5）线程；

（6）资源分配；

（7）处理机调度；

（8）内存管理；

（9）磁盘分区和管理；

（10）I/O控制；

（11）文件系统管理。

4. 数据结构

（1）算法的时间复杂度和空间复杂度；

（2）线性表；

（3）栈；

（4）队列；

（5）树的基本概念；

（6）二叉树；

（7）图的基本概念；

（8）图的遍历以及图的生成树；

（9）查找，包括顺序查找、二分查找；

（10）排序，包括插入排序、选择排序和交换排序。

5. 数据库

（1）数据库的发展历史；

（2）关系型数据库；

（3）字段，关键字；

（4）表；

（5）索引；

（6）触发器；

（7）存储过程；

（8）作业；

（9）事务以及事务的提交和回滚；

（10）游标；

（11）SQL语言；

（12）掌握一种关系型数据库的使用，例如SQL Server， Oracle；

（13）数据备份和灾难恢复；

（14）数据导入导出；

（15）权限控制；

（16）数据库设计初步。

6. 软件工程

（1）软件工程的概念；

（2）几个知名的软件开发模型：瀑布模型、螺旋模型、增量模型等；

（3）需求分析；

（4）软件设计的基本原理：模块化、抽象、耦合、内聚；

（5）程序流程图；

（6）软件测试的基本概念；

（7）单元测试；

（8）集成测试；

（9）功能测试和性能测试；

（10）白盒与黑盒测试；

（11）评审；

（12）配置管理；

（13）CASE（计算机辅助软件工程）；

（14）了解一点ISO9000和CMM的知识；

（15）了解一点知名软件公司的软件研发流程，例如微软公司。

7. 网络

（1）几种常见的网络拓扑结构：总线型、环形、星型、树型、网状、混合型等；

（2）OSI参考模型（七层协议）；

（3）TCP/IP；

（4）以太网；

（5）常见网络设备，例如路由器、网桥、中继器、网关等；

（6）广域网；

（7）熟悉一种操作系统上的网络配置和常见问题分析。

对于以上这些基础知识点的选择，也是仁者见仁、智者见智的事情，所以只能给大家一个参考。

2.3 测试工作需要的基本品质

2.3.1 积极

因为工作关系，我与一些公司负责招聘的经理保持联系，有一次，我去拜访一位部门经理，在谈到他期望什么样的员工的时候，他说，除了基本的技术要求外，他需要积极的人。新技术可以再学，而状态不积极则不容易改变。并不是只有这位特定的经理这么说，他的这个要求具有普遍意义。

员工积极与否，对于工作来说完全是两重天。如果你只是一个“机器人”，等待着老板的下一条指令，这会很麻烦，你会比较难得到老板的赏识。公司不是学校，老板没有耐心对你做如同管小学生一样的管理。如果老板真的这么做了，他的团队也不会真正强大起来。

可能很多人都会认同我的观点，认为积极是一项重要的品质，但是会有两个疑问：

（1）自己的状态是积极还是不积极呢？

（2）怎么才能做到积极呢？

只有发现了问题，才有动力去解决问题。有的时候，我们作为一个个体，因为“当局者迷”，往往意识不到自己的状态不积极。当老板提出这方面的劝勉时，自己还会说：“不会吧，我挺认真的，我在组内至少是一个中等吧……”。好，那让我们静下心来回想一下自己的言行举止，看看哪些容易被判为“不太积极”。先从表面上来说，例如项目组比较空闲的

时候，你一边磕瓜子一边看小说，当然不如看专业书或者准备后期工作的同事积极；例如掐点上班，生怕早上班一分钟。再看工作中，比如你虽然能按时完成工作任务，但总是拖在最后；提交工作报告，在最后完成的几个当中又有你；开会，能不发言就不发言；自己负责范围之外的事情，很少过问。诸如此类，等等。

一个人被判定为积极或者不积极，是老板和同事根据长期印象来判断的，你不会因为一天的积极而得到一个“积极”的评价，也不会因为一件事情表现不好而得到一个“不积极”的评价。我根据自己的体会，提出下面一些建议，大家如果觉得合适，可以试一试，长期坚持应当能得到好的效果。

1. 把自己的工作做好

早上班、晚下班只是一个表面的行为，关键还是要做好自己的本职工作。多考虑，深入分析，把工作完成得让同事信服，让老板放心，这是得到良好评价的根本。要想在“完成”前面加上“出色”两字，真是要下一番功夫的。

2. 多考虑与自己工作范围有衔接的“公共区间”

完成自己的工作任务，这是很多人认为“本分”的事，所以这一条做起来不是太难。如果你只是局限于自己的那“一亩三分地”，就有些保守了。例如，如果你是一位测试工程师，你测试模块B。模块B在整个系统中不会单独存在，你把怎么测试模块B考虑清楚了以后，你还要看看模块B和模块A、模块C的“公共区域”是什么，把这个“公共区域”也纳入到自己的工作当中来。宁愿这个“公共区域”有多个“婆婆”在管着，也不能出现“三不管”地带。如果你能做到这一点，老板一定会暗自赞许：嗯，还不错，状态积极。

3. 与同事分享自己的经验

如果你在工作当中是一个有心人，你肯定会有一些体会，积累下来，

就是经验。大家同在一个团队中工作，如果你能与同事分享你的经验，你同样也能得到他人的经验，这对个人和团队都是一件好事情。这样做也是一种积极的表现。

4. 多发言

开会的时候，如果只有主讲者在说话，是一件挺没有意思的事情。特别是技术类的工作，会议的目的都是要集思广益，而不是听一个人做报告。只有经过讨论甚至是争论的结论才会更正确、更有价值。你多发言，就是对团队多做了贡献。从个人方面来看，多发言也是一种积极的表现。

5. 多参与团队活动

公司或者小组组织爬山、郊游、聚会等，我们要积极参与。的确，你不参加也无可厚非，别人不可能强迫你去，但是，如果一直是这样的话，你就会给人留下对团队活动不积极的印象。更重要的是，团队活动是一种很好的交流方式，因为工作当中的交流都是比较正式的，而在团队活动中，你更容易接触到同事在生活中的一面，你不参与就把沟通的好机会白白浪费了。

只要你有心，慢慢做，你会变得积极的。你积极的举动会让你得到更多。

2.3.2 良好的团队合作能力

对于上班族来说，带薪的年假总是令人期待的。好不容易盼来了年假，轻松愉快地踏上了旅程,这个时候，你希望在欣赏美丽的风景的时候，不停地接听工作方面的电话吗？一个电话，就可以把你从悠闲的放松中拽到紧张的工作中。如果在你休假的时候没有人接手你的工作，没有人知道你这一块的事情，你就等着电话来找你吧。这只是团队合作中出现的负面的一个小细节。

现在“团队合作能力”几乎出现在每一个招聘启事中，也越来越多地

出现在大家的简历中。当你在自己的简历中写下“具备良好的团队合作能力”的时候，你是否认真考虑过，自己真的如同自己写下的一样，团队合作能力不错吗？而当你工作在一个团队中的时候，你或许对自己的团队合作能力毫不质疑，甚至不大在意。让我们稍微静一静，花几分钟的时间考虑一下，怎么做可以算是具有良好的团队合作能力呢？

就我个人的经验来看，我们可以从下面几个方面来衡量：

1. 有意识地关注项目进度和组内情况

做好个人的本职工作，大家都是没有异议的。而对一个团队合作能力不错的人来说，他（她）还会有意识地关注项目进度和组内的情况，例如现在实际进度是否符合计划的要求？潜在的风险是什么？他人的进度如何？有什么新的变化吗？我们每个人负责的那一块工作是整个团队的有机组成部分，随着项目的进展和团队的动态变化，我们需要做一些动态的调整。只有了解了整个项目和团队的情况，才能更好地实现团队合作。

2. 愿意共享经验，也善于从他人那里学习

哪怕是专家，每个人都只能做项目中的一部分工作。如果每个人都乐于与大家分享各自的经验，不管是以前积累的还是从当前项目中得到的，大家就会进步很快，从而更快更好地完成项目。每一个人都应当抱着开放的心态，乐于从他人那里学习。如果大家既不分享，也不从同事那里学习，保守地做自己的工作，整个团队的效率就下降了。

3. 以团队利益为思考出发点，不过多计较

在一个团队中，如果你要计较绝对的公平，会有很多麻烦。例如，两个模块，一人一个，你可能会说，呀，我的工作比他多一点，一点点，他是0.999，我是1.000。如果你是leader，你又能怎么操作呢？这只是一个比较绝对的例子。更多的情况会在追加工作量的时候，你容易感到不平衡。例如新增了一个小功能，给谁来做都是追加工作量，所以这个时候，要以团队利益为出发点。对于团队来说，需要有人来做这件事，如果指派给了

你，你就去做好了。

4. 互助

如果团队中有同事跟不上进度（因为身体不适，或者遇到了技术难点等），而你又提前完成了自己的工作，你可以去帮助他。这对你来说虽然带来了额外的工作量，但会增进你们之间的友谊。更重要的是，你为团队按时完成任务提供了保障，这是很有意义的。这种自动“补位”的精神，能显现出一个人的团队合作能力。

5. 虚心，认同他人，尊重他人的工作

无论我们自身的工作做得如何，我们都应当保持虚心，从内心上认同他人，尊重他人的工作。这一点会在你处理一些矛盾的时候表现出来，如果你认同他人，不管是什么事情，都可以协商解决。如果不是这样的话，即使一点小事情也会越来越麻烦。

6. 批评的时候只对事，不对人

这是一种在团队中工作的技巧。每个人都会犯错误，而且每个人都不能忍受攻击，所以当你发表意见的时候，如果是批评的，就要表现出善意，要对事不对人。如果你真正为项目、为团队考虑的话，就请这么做。对他人的不妥的批评容易瓦解一个团队。

7. 友好、宽容地对待新同事

回想自己当新兵的时候，你就会知道新同事需要时间来逐渐适应新的环境，所以，请友好、宽容地对待新同事，不要以老同事的标准来要求他（她）。另外，如果你给予新兵力所能及的帮助，你会得到一个朋友☺。

如果你是一位在校学生的话，你一样可以锻炼自己的团队合作能力，而且也需要这么做。

2.3.3 耐心和细心

在国内软件行业刚刚兴起的时候，有一些老板对软件测试工程师除了

在计算机专业方面之外还有一个要求：女性，或者写道：“男女皆可，女性尤佳”，因为他们认为只有女性才能真正有耐心来做好测试，这个想法当然不对，但也可以看出软件测试这份工作对耐心的高要求。

测试工作究竟在什么地方体现出对耐心的要求呢？我们可以从软件测试的各个阶段来说说。

首先是参与软件需求的确定和评审。我敢肯定，测试工程师是除了需求规格说明书的作者之外阅读这份文档最认真的人，基本上就是字斟句酌，从根本上把握软件的“蓝图”。不这样做，测试势必会做得粗糙。同时，测试工程师也要尽量发现需求说明书中的问题，根据软件工程的理论，问题发现得越早，解决的成本越低。要彻底把握需求，反复读个几遍、十几遍是很正常的，如果没有耐心，怕是做不到。

接着是软件设计和编码阶段，软件测试工程师一般在这个阶段设计测试计划和测试用例。在设计测试用例时，最大的挑战是如何考虑得更周全，尽可能多地提高测试覆盖率。如何考虑得周全是一个渐进的过程：做完第一遍，自己重新看看，审核一下，或者请别人帮忙看看，发现不足之处，做改进和补充，出第二稿。再次重复这个过程，出第三稿……。没有耐心的朋友坚持不了几个回合。

测试执行阶段更是考验人的高峰期。假设你有了400个测试用例，根据整体安排，一周执行一遍，周期为两个月。在这两个月内同一件事你要重复做8遍，你能不能坚持下来☺？这只是一方面。另外，我们还要在这样近乎枯燥的重复工作中发现bug，不然即使坚持了下来，却没有发现bug，是没有什么意义的。还有，发现了bug后，还要甄别是不是bug，详细描述重现的步骤，并做一定的原因分析，也是需要耐心做支撑的。

因为测试工程师是一个找错的工作，在一些做开发的朋友眼里看来就是“挑刺”，这样我们在与开发人员进行交流的时候，就要更有耐心一些，以使让他们理解问题所在。如果简单行事，会不容易被对方所接受。而一

旦有了误会，沟通成本就大了。

在细心方面的要求，也是和耐心一样。在软件研发和测试的各个阶段，只有足够的细心，我们才能发现更多的bug。

耐心和细心是可以后天培养的，任何人不可能在这两方面做得尽善尽美，耐心和细心的形成也是一种渐进的过程。只要我们自己有意在这两方面去锻炼自己，会做得越来越好的。

03

第3章 如何找工作

现在因为社会竞争激烈，无论是对于刚毕业的朋友，还是对于想换工作的朋友来说，找工作都是个沉重的话题。虽然有的人因为天资高、关系硬、运气好等等，左右逢源，但对于我们大多数的人来说，毕竟还是要经历炼狱般的找工作的过程。我们能找到什么样的工作，当然主要取决于我们个人的能力，而这其中也有一些好的经验和方法，值得分享和借鉴。在本章中，我会和大家一起仔细地讨论找工作过程中的每个环节。我希望我们每个人在找工作过程中都能更顺利一些，少走一些弯路。

3.1 工作的种类

在开始讨论如何找工作这个话题的时候，可能有的朋友会问：“我的简历怎么写？”，或者“面试中需要注意些什么？”，或者有的更单刀直入地问：“你能帮我找一份工作吗？”。哈哈，我没有那么大的能耐，没有办法直接为你找到工作，也不可能越俎代疱这么做。简历和面试方面的注意事项都是具体操作上的技巧了，在讨论它们之前，也就是在我们投简历之前，我们需要想一想，自己要找一份什么样的工作？

计算机软件行业是一个大行业，现在随着软件业的快速发展，行业内部有了细分，提供的职业机会也是多种多样。下面我列举一些，希望对你有用：

1. 软件开发工程师

相信大家对这个职业是耳熟能详的，软件开发工程师是软件业最传统的一种职业，一般来说软件工程师就是指软件开发工程师。如果你想应聘软件工程师，你需要有比较好的编程功底，逻辑思维要强，有较强的学习新知识的能力。软件开发工程师有多个方向，如网站开发，数据库开发，数据仓库开发，C/S架构的程序开发，大型平台的二次开发等等。

2. 软件测试工程师

测试工程师的角色是站在用户的立场上对软件进行测试，找出软件的错误，使软件产品能达到发布的标准。它的基本要求是计算机专业知识扎实，耐心，细心，沟通能力好。

3. 程序经理

在大的外企软件公司，程序经理负责需求规格说明书的编写，并推动整个研发团队的工作。它的要求是英语要很棒，计算机基础知识扎实，组织能力强。注意，这个经理不是官职，只是一种职业名称，所以即使是应届毕业生也可以申请这种职位。但在国内的公司，程序经理大多是作为系

统分析员，不但要写需求规格说明书，还要写系统的设计，这些人员都是高级程序员提升上来的，是一种管理职位。

4. 系统管理员/网络管理员

例如，一个大型的网站需要工程师来做日常维护，一个公司的内部网络也需要有人来维护。如排除小错误，系统检查，数据库备份，硬件增减，账号管理，安装操作系统等。这种职位的要求是计算机基础知识很宽泛和扎实，熟悉常见网络设备和软件，有耐心。为什么要有耐心呢？因为系统管理员的事情会很杂，如果没有耐心，怕是做不长久。

5. 系统部署工程师

举一个例子，一个公司或者工厂向微软购买了一整套成熟的内部办公软件，这就需要系统部署工程师来把其中的各种软件安装并调试好，方便客户的使用。如果你想做这个职业，除了计算机知识外，工作经验也需要一些。如果你有一些大公司的认证，也是很有用的。如果是做微软产品的部署，一般都要求MCSE证书。另外，可能你也看出来了，这种职业出差的可能性比较大，不愿出差的朋友要慎重考虑。

6. 文档工程师

一般只有大公司才有这样的职位，主要是编写随着软件发布的技术文档。它需要你有计算机专业背景，并且文字编辑能力较强。

7. 质量保证工程师

这个角色的任务是监督整个软件研发流程，包括开发和测试，以确保研发工作科学地展开。要从事这种职业，一般都要求很熟悉软件工程理论，并有几年的工作经验，因为如果没有经验很难去有效地监督别人。

8. 美工

软件行业内还有美工？是不是我说错了？不是的。随着用户对软件品质的要求越来越高，界面设计方面的工作慢慢由专业的美工来担任，他们

设计出图片，将其提供给软件开发工程师。

9. 软件本地化工程师

其实就是做翻译工作，你需要精通一门外语，并对这门外语中的计算机方面的词汇比较熟悉。

这里只列举了一些常见的职位，软件行业的管理职位我没有列出来，因为我想管理职位属于发展性的职位，大家有了几年的工作经验之后自然能够了解到，这里就不做讨论了。

如果你选择软件测试工程师为职业，那么，欢迎你！本书的目的正在于希望能带你迈进软件测试的大门。

3.2 什么是自己的优势

在我做面试的时候，我通常会问应聘者一个问题，你为什么想应聘这个职位？记得有一位朋友，以前一直做开发，现在来应聘测试工程师，他说，开发太辛苦了，所以不想干了。还有一位朋友，也是来应聘测试工程师，她说，大家都认为女生适合做测试，我就想自己可能适合，所以我来试试。

他们的回答都不能让我满意，因为他们在选择一个职业的时候，在分析一个职业是否适合自己的时候，没有认真地想过自己的优势是什么，而是怕辛苦或者人云亦云，这样的风险很高。

现在的竞争很激烈，哪怕是一个普通职位，都会有多个人同时申请，要想脱颖而出，的确是要有个人优势。从个人的角度来讲，如果在分析自己的实力后看到了自己的优势，会对自己的职业发展有积极的影响。我有一位朋友，毕业后做了一年的软件实施和开发，觉得软件开发并不是自己的强项，正好当时公司组建测试组，内部招聘一个测试组长，他就申请转做测试。因为是内部调转，而且他以前的工作以及态度也都得到了认可，

所以相对比较容易。积累了几年的测试经验和测试管理经验后，他转到了一家外包公司，之后又进入一家大型外企做测试，掌握了成熟的软件工程的思想和测试思想，现在他出版了一本测试方面的书，也成为了一家公司测试部的经理。他说："如果我一直做开发，我只会是一个普通的开发人员，但是，我想，我细心、耐心、温和，与他人沟通比较顺畅，加上计算机基础知识扎实，而且有一定的开发背景，逻辑思维也不错，所以如果我做测试，可能会是一个不错的测试人员。经过实践，的确，在测试这个方向上发展更适合我。"。每个人都有优势，且各不相同，需要具体分析。

怎么认清自己的优势，也是一个问题。当局者迷，看清自己也真不是件容易的事情。有的朋友对我说，我不知道自己适合做什么。这是个大问题，这说明还没有看清自己。我建议你先把自己的想法写下来，然后说给几个朋友听，看看他们的反应。有的时候可能会有讨论或者争执，但这正是我所希望看到的。经过这样的讨论，可能你对自己的认识会逐渐清晰起来。当然，对自己的优势的判断是分阶段的、渐进的，过了一个阶段后，我们自己会发生变化，对自己的认识也会发生变化。

3.3 取舍

工作如此重要，所以往往寄托着我们很多的希望，例如我们希望它是自己喜欢做的，薪水不错，工作环境很好，有前景，在自己期望的城市工作，等等。这些期望都是合理的，是人之常情。同时，我们也会看到，现实又往往要我们做一些取舍，一份"十分满意"的工作是难求的。

例如，应届毕业的朋友有的时候会在期望的工作和户口之间徘徊。能解决户口的一般都是国有企业，有的甚至不景气，报酬很低；不能解决户口的公司，可能所做的工作与专业会更对口一些，能学到很多东西，但没有所在城市的户口，相对来说也是不稳定的。怎么选择呢？这当然要看你

自己的想法。如果你认为户口重要，你愿意为这个而牺牲其他的，无可厚非。如果你认为工作的实际内容比户口更重要，也对。只要是你自己的决定就好。

加薪和转行往往不能两全。我接触过一些应聘者，以前做开发，对软件测试不了解，没有测试经验，希望转做测试。但他们提出来的薪金要求却不低，是在自己以前薪水的基础上又提高了一些。但是，你想，如果你真心要转行，就要有一种谦虚的心态，认为自己不懂，希望能找到一份工作来实现自己的转行计划，在薪水上没有过多的预期，这在现实中才更可行一些。转行又加薪的好事有是有，概率很小。虽然你在行业A内有几年经验，但在行业B中来看，并不能构成薪水增加的理由。甚至，有的时候，你需要为你的转行付出降薪的代价。

有的时候，在自己喜欢做的工作和自己所能得到的工作之间也会有取舍。要做自己喜欢做的事情，这是大家公认的道理。但是，如果现实中暂时是只能得到另外一份工作，那就不如先干着。只要上班了，经验就会得到积累，以后再换会容易一点。例如，你想做软件测试工程师，而公司却认为你没有经验，但看中你年轻，无牵无挂，出差有保障，所以可以接受你做一名软件实施工程师，就是要你出差去客户那里安装、调试软件并做相关的培训和维护。虽然工作内容离软件测试远了一点，但是如果没有更好的机会，就先做吧。积累了经验之后，内部转做软件测试会容易一些。

做一些取舍其实是在现实条件下实现自己的利益最大化，让自己灵活一点，离自己的目标更近一点，而不能让多个矛盾的条件同时存在，互不相让，影响自己的决策。

3.4 尝试

有的时候，我会收到读者朋友的来信，跟我讲他们工作方面的苦恼，

其中最难以做出答复的是：“什么是适合我的工作？”。这个问题，只能自己才能给出答案，他人哪敢妄下定论？

著名电视节目主持人杨澜，大家可能都知道。她离开中央电视台后，几经坎坷和曲折，到今天，在商业上也算取得了成功。记得有一篇采访她的文章，她在说到一个投资的案例时表示：“我们也不知道对不对，但是只要认准了，我们就一头扎进去，去试。半年一年后，如果错了，我们就撤出来，我们还年轻，试得起。”。有的时候，找工作也需要有这种心态，要去尝试。

举个例子来说，现在测试行业行情不错，很多朋友有点心动，既想试试又不大敢尝试。不管你从报道中看到的或者听到的软件测试是什么样了，只有在你亲身接触过这个行业之后，你才会知道究竟自己适不适合。所以，我想，如果你觉得自己适合某一行业，你就要勇敢地去尝试，这远比在脑袋里想来想去强得多。

但是，尝试是需要代价的，特别是现在，找个工作不容易，尝试的成本就更大了。我们既要有勇气去尝试，又要尽可能地降低尝试的成本。不要做盲目的尝试，因为那只会让自己更加窘迫。

首先，确定自己是否符合目标职位的基本要求。例如，软件测试工程师，需要有宽泛和扎实的计算机基础知识，有耐心，工作细致。如果有一定编程能力的话，通过面试的概率要大一些。这些只是成为软件测试工程师的基本条件。你可以参照这些条件来看看自己，如果大部分都不能满足，我劝你还是不要去尝试，因为你获得软件测试工程师职位的概率比较低，求职的成本就高了。而或许你去尝试另外一个职位的成本要低一些，成功的概率要大一些，因为每个人都有自己的长处。

其次，开辟另外一些尝试的途径。还是以软件测试工程师这个职位为例，你可以买几本测试方面的书看看，在阅读的过程中你会逐步增进对这个职业的了解。如果财力允许，你可以参加一个测试方面的培训班，培训

中会有理论课和一些实战练习，这会让你对软件测试有比较深入的了解。虽然花了一些钱，但它会让你在选择职业道路的时候风险低一些。

如果以上两个步骤你都做过了，仍然觉得自己挺适合做这份职业，这个时候就要下定决心，尝试着去找一份这方面的工作。相信自己，只要自己是真心的，会有收获的。

3.5 销售自己

在备选的工作当中，“业务员”或者“销售”是一份让很多人望而生畏的工作，觉得难度太大。而从某种意义上来说，找工作，就是把自己销售出去，并且还要是一个好的去处。要当好自己的销售员可不容易。

首先，请写好自己的“产品说明书”。在写简历的时候，要把自己的特点写出来，这样比较容易引人注目。除了基本信息和工作经历之外，需要通过简历明确地告诉别人，我们的优势是什么？擅长什么？对简历进行筛选的人没有那么多的时间在你的简历中来回寻找关键信息，让他们一目了然是最好的。这就好像在超市里，顾客拿起一种产品，如果花了半分钟的时间还不知道这个产品究竟有什么特色，他可能会放下它，选择另一产品去了。

接着，拓宽渠道。人才交流会、招聘网站、招聘公司的网站都是重要的信息渠道，自然不能遗漏。同时，老师、同学、朋友的介绍也是一个辅助渠道，说不定也有一些有用的信息。所以，如果你在找软件测试方面的工作，请大方地告诉你圈子里的人，这样如果有了机会他们自然会想起你，例如他们的公司需要招聘人员的时候。

然后，注重面试。面试是一个宝贵的机会，雇主零距离观察你，你可以充分地展示自己。面试当中，最重要的是自信。你不知道面试官会问什么问题，只有保持自信才能应付得过来。另外，现在一般在面试结束的时

候，面试官会礼貌性地给应聘者一个提问的机会，我建议你抓住这个机会，问一些自己想了解的问题，例如公司的发展方向，这个职位的工作内容等等，不要以为问问题会显得不恭敬。通过问问题，你可以得到一些以前不知道的信息，有利于你做出一些基本的判断，例如这个公司是否适合你，这个职位是否适合你。

最后，要注意做好分析和总结。例如，在一段时间内，投送了多少简历，有多少面试机会，面试中有什么做得不够好的，等等。经过分析和总结，我们对自己适合找什么样的工作和如何能更好地找到工作会有进一步的了解。

总而言之，就是保持销售的意识，努力去找工作。

3.6 简历的注意事项

我相信简历的重要性大家都知道。首先在一家公司面前亮相的往往是你的简历，而不是你自己。简历代表着你，面试官根据简历来决定是否给你面试的机会。如果连面试的机会都很少，那么找到合适工作的机会就更少了。简历的基本要求大家都知道，这里就自己的一些经验说几点需要注意的地方。

1. 个人主要信息要醒目

个人的主要信息都要以醒目的方式标出。什么是个人的主要信息？例如姓名，性别，年龄，学历，联系方式，工作年限，擅长的技术点，项目经验等。不能让筛选简历的人到处找你的特点，而应当是让这些特点主动地抓住他们的目光。例如，如果一个公司要招聘一位测试工程师，要求懂技术A，而却很难从简历上发现你有与A相关的工作经验或者学习经历，你入选的概率会大吗？可能面试的机会都会很少。我们可以通过一些简单的方式来着重强调我们的主要信息，例如把字体加粗，或者让它们单独占一

行，或者用其他方式。我曾经接触过一些简历，需要我从头至尾反复看好几遍才能大概看明白，需要我去总结“中心思想”。如果工作忙没有时间的话，这样的简历往往就被忽略了。

2. 着重描写工作经验

现在公司衡量一个人是否合格，他的工作经验占据很重要的权重，这在软件行业尤其明显。因为工作经验很重要，所以请在简历中开辟一个专门的区域去详细叙述它。项目的名称、起止时间、所用技术、你的角色和职责等，都要说清楚。一旦公司在你的工作经验中发现了他们所期望的或者相近的，你得到面试机会的概率会比较大。没有工作经验或者经验缺乏的朋友也不可忽略这一部分，自己的实习项目、学习项目都是可以写的。如果一点相关内容都没有，简历会显得很单薄，没有竞争力。

3. 描述对某技术或某职业的理解

如果你对一门技术或者一个职业有自己的深入的理解，可以写出来。这种理解因为是自己归纳的，所以往往比通常意义上的“自荐信”还要有效果。例如，如果你想申请一份测试工程师的工作，而你对这个职业有自己的理解，你可以在简历中安排一个小段落来叙述你的看法。这一部分并不是简历中必须的，有则锦上添花，没有也没有关系。只有对要陈述的对象有深入的了解的时候，再去写才比较好。如果不熟悉，反倒是搬起石头砸自己的脚。所以，做之前，慎重考虑一下。

4. 正式

简历要给人一种正式的感觉。在某种意义上，简历就是一种公文，是你呈给公司的公文，正式为好。这就需要在排版上多加注意，字体大小、字体颜色、段落距离等都要符合一般的要求，而不是随心所欲。

5. 准备多份针对不同职业的简历

如果你申请两个不同类型的职位，你最好准备两种不同版本的简历。例如，如果你现在认为开发或者测试工作都可以，那么你不要尝试着仅以

一种格式的简历“打天下”。这两种工作的侧重各有不同，当然需要你在不同的简历中有所侧重。否则的话，可能招聘测试工程师的公司认为你不适合测试的工作，而招聘开发工程师的公司又认为你不适合做开发，这是件糟糕的事情。

6. 简历文件名要包含姓名和关键信息

很多求职者的简历的文件名只是两个字“简历”，这对招聘公司的人事专员来说有点麻烦，因为他们很难把相同文件名的简历迅速分开，也容易造成文件的相互覆盖。如果你的简历的文件名是“简历_你的名字”，这样就方便多了，如果能再进一步，例如“简历_你的名字_申请xx工作”，就更好了。

以上这些都是我的一些体会，供大家参考。

3.7 笔试

现在笔试越来越受到重视，尤其是在技术类岗位的招聘中。现在求职者甚多，简历中又多少有一些修饰，如果只是在面试过程中问问“你会不会xx？”或者“使用过xx吗？”，有些过于简单，无法知道候选人的实际水平，很难挑选出合适的人员来。这样，笔试对招聘单位来说就很有用。不管你说你会什么，做做题就知道了。一般来说，笔试会是第一关，招聘单位会根据你的笔试成绩决定是否给你面试的机会。这样，对于我们求职者来说，过好笔试这一关就变得很重要了。

在找工作期间，我们在等待面试通知的同时，可以花一些时间去有目的地复习一下功课，以期顺利通过笔试。如果想找一份软件测试的工作，笔试一般有以下内容：

（1）开发题。在规模稍大一点的软件公司中，对测试工程师都有开发能力的要求。如果你一行代码都不会写，很难让人相信你的测试能力有多

强。开发题不会很难，主要是考察你的逻辑思维能力。对这一类的题，你需要复习一下具体编程语言的语法，还有一些数据结构方面的知识，例如排序算法等。

（2）测试题。公司会通过一道或者几道测试方面的题，考察你的测试能力。因为发散性是测试思维中很被看重的一种能力，所以你在做这类题的时候，要尽可能多地罗列出你的想法，而不要只满足于一些简单的考虑。简单的考虑大家都会，如果你也仅限于此，怎么能脱颖而出呢？

（3）英语题。英语题大多是将一段英文翻译成中文，科技类的居多。如果你一段时间没有使用英语了，温习一下，避免看不懂简单的单词。

短期的复习当然不可能让你的水平有一个飞跃，但可以达到我们的目的，避免因为长时间没有接触而生疏了，导致笔试失利。

在笔试的过程中，有一些细节应当被注意：

（1）字迹要整洁。因为字写得是否端正，有的时候会与你的面试态度和工作态度挂上钩。不要求你写一手好字，但至少要整洁，没有辨认的困难。

（2）不要交白卷。如果试题的确很难，你仅能答出一小部分，那你也应当把自己知道的题做完，然后和笔试组织人员说明，其他题的确不在自己的能力范围内。如果你交了白卷，一是招聘公司会认为你水平很差，一道题都不会；二来可能还会认为你是轻言放弃的人，对你的工作能力产生疑问。

（3）可以申请更换笔试题。如果你发现一套笔试题不适合你，过难或者和自己的知识结构有很大的偏差，你可以有礼貌地去交涉，说明情况。例如，如果你发现中级测试工程师的题很难，你可以申请改做初级测试工程师的题。哪怕你被拒绝了，也没关系，毕竟尝试过了，而我相信很多公司能接受这种申请。

（4）试卷上要留下姓名。笔试的人会很多，如果你大名都不留一个，

因此造成的偏差和延误只能由你自己承担了。

另外，如果你事后发现自己有一道题应当做出来而当时却没有做出来或者做错了，你可以通过邮件的方式再次提交答案。这样做或许没有作用，但也表明你很诚恳，在意用人单位的笔试，在特殊情况下说不定有一些效果。

本书的第5章专门讨论笔试题，选择了一些典型的测试工程师职位的笔试题，对每一道题都做了分析，并给出了参考答案，欢迎阅读。

3.8 面试

3.8.1 不必过于重视面试技巧

面试的重要性不言而喻。如果通过了面试，你就能得到职位；通不过，就没戏。在一职难求的今天，面试是让每一位候选人期待而又有点心里没底的事情。大家都希望自己在面试中表现良好，一批传授面试技巧的书也应运而生。我不反对求职的朋友看一点这方面的书，它们多少会有些帮助，特别是通过学习一些失败的例子来避免自己犯同样的错误。但是，我想对大家说一句的是，不必过于重视面试技巧。

例如，有一个传播很广的例子，说一家大公司招人，面试结束后大家觉得问的问题很简单，都感觉良好。可是，实际上，玄机在等候室和面试室之间的走廊上。在从等候室到面试室的中途有一把扫把倒在地上，暗中有人观察，谁把它扶起来，谁就是胜出者。这个例子中的面试官的本意是通过一种特定的场景，去观察候选人的真实品质。如果你看过这个例子后，就在面试的时候四处找扫把或者其他什么东西，就有点牵强了。那种有明显做作痕迹的面试技巧不会有什么作用，有的时候还会起反作用。

我曾经面试过一位候选人，她来应聘测试工程师。她虽然有几年的测试工作经验，但是我发现她的编程功底较差，逻辑能力不强，测试方面的

思维能力也不突出，工作能力与她拥有的几年工作经验不相匹配。但我认为这没有什么，因为人各有特长，只能说明她暂时不适合我们的这个职位，所以面试结束后，我和人事专员还是很客气地送她出公司的门。每个候选人到我们公司来面试，我们会给候选人拿一份内部刊物，让他们在等待的时间里翻阅，这样候选人既打发了等待的时间，又增进了对我们公司的了解。在我和人事专员送她出门的时候，她对人事专员说了一些对我们内部刊物的评论，特别指出了错误。从她的神情来看，我猜她是受到某一本介绍面试技巧的书的影响，想通过某种“旁敲侧击”来提升我们对她的评价。这当然不会产生什么效果。

相对而言，候选人在面试中处于弱势地位，面试官总是处在一个制高点观察你。面试技巧一类的东西，只能心领神会，融合在自己的言行中，而不能刻意表露。而且，要注意的“面试技巧”一多，你的注意力可能就会分散，而不能集中精力回答面试官的问题。所以，我的建议是，不必过于重视面试技巧，而应当与面试官坦诚相见，展现出自信而真实的自我，那是最好的。

3.8.2 如何陈述工作经历

“简单介绍一下你的工作经历。”，这是很多面试者在面试中遇到的第一个问题。因为现在招聘单位都很看重工作经验，尤其是软件行业，所以我想有必要在这里单独说一下如何在面试中陈述自己的工作经历。

在“说道理”之前，我先举几个小例子：

候选人A，有几年的工作经验。他对上面的问题的回答是：“简历上都写着呢”。他心想，我的简历上写得清清楚楚，干嘛还问我？

候选人B，大学刚毕业，没有工作经验。他有点紧张，他说：“我没有什么工作经验，很抱歉。”，然后就没有了下文。

候选人C，有多年工作经验。但不知是他简历中写错了，还是他叙述错

了，他所说的和简历中写的有点出入，特别是时间上。例如，在简历中某项目是某年上半年结束的，他却说成了某年下半年结束的。

大家可能从我的字里行间能感觉出来，这三位候选人的做法都欠妥当。那么，他们在什么地方做得不对呢？在面试中叙述工作经历要注意些什么呢？

首先，态度要诚恳。正如候选人A所嘀咕的，简历上都清楚地写着呢，干嘛还要问我？实际上，很多面试官都是拿这个问题来做一个引子，他会根据你的叙述再进一步问你一些问题，例如当你说到某个项目，面试官可能就会问一些细节问题。如果你一句“简历上都写着呢”就回答完了，不但给面试官留下了“锋芒太盛”的印象，而且打乱了面试官的面试节奏，对候选人不利。即使简历上写了，再复述一遍又何妨？毕竟是你在找工作，而不是面试官。

其次，要有备而来。在面试前，自己可以先练练，让朋友当面试官，也可以自己一个人说说，这样在面试的时候会更流畅。准备的时候，要多准备一些，特别是具体的项目情况，一般面试官都会进一步就具体项目问几个问题。

再次，要有自信，扬长避短。如候选人B，虽然没有什么工作经验，但可以这么说：“我没有什么工作经验，但是我在大学里跟着老师做了几个项目，这些项目都很有价值，我想我的这段经历与实际工作经验一样有价值，对我来说很宝贵，因为我学到了很多东西”，然后就可以详细叙述自己的项目经历。这么一说，面试官会对你感兴趣，对你留下较深的印象。

最后，要准确，说的要与简历上所写的匹配。如果你和候选人C一样，自相矛盾，即使你不是在说谎，面试官也会暗自猜疑：“这个人是不是在说谎，不然他自己说的怎么会与简历上的不一致？”。如果面试官真的这么判定你，你就不会有被录用的机会。造成这种失误的原因可能只是你对好几年前的事情记得不是很清楚，加上没有准备，在复述的时候时间有点

错位，这样是很可惜的。

刚刚毕业的朋友们，当你在应聘测试职位而被问到测试经验的时候，不要简单地说："很抱歉，我一点测试经验都没有"，这句话哪怕说得再诚恳，面试官也会觉得爱莫能助。你可以准备一下在大学期间的毕业设计、课题设计、实习、组织活动等等的经历，然后有条理地说出来，这样可以让面试官感觉到你虽然没有直接的工作经验，但是那些设计和实习也可以折算成工作经验，这对你有利。

面试官以让候选人介绍自己工作经历来开始面试，然后会就一些你经历过的具体项目问一些相关的问题，如果你能回答流利、顺畅，会对你的面试过程以及结果产生积极的影响。

3.8.3 技术面试

在某种程度上来说，技术面试重要到能够决定你是否被聘用。在技术岗位方面，在个人品德没有问题的前提下，招聘公司对技术是最关心的。

我现在并不能给你分析具体的面试题，因为与笔试题相比，面试题千变万化，不同的公司有不同的技术方向，即使是在一个公司内，技术面试题也会因为项目、岗位、面试官的不同而不同。下面我说一些在技术面试中需要注意的地方，和大家一起做个交流。

不要对一些问题只给出简单的"会"或者"不会"的回答。在技术面试中，比较常见的问题是："你会不会技术A？"或者"你懂不懂技术A？"，根据我们的思维定式，看起来好像回答"会"或者"不会"就可以了，其实不是这样。面试官是想从这个问题开始，尽可能多地了解你在技术A方面的情况，如果你能对你的"会"再作一些说明就更好了。例如，你可以说，"会。我比较熟悉技术A，因为在去年的一个项目中，技术A是主要技术之一。我作为项目组成员之一，在半年的项目开发期内一直使用它。"。如果你对技术A不大熟悉，你也可以做一个简单说明："有点抱歉，我没有专

门的技术A实战经验。但是，我从一些技术文章中看到过相关的介绍。我想，根据我的项目经验，只要有机会，在实际项目中学习，我应当可以比较快地掌握它。”。这里有一个基本准则，所有你所说的都应当是真实的，例如你说你有实力很快掌握它。我在这里不是教你撒谎，只是希望你向面试官表达得更充分一些，假话说得再漂亮也没有用。

不要以为所有的技术你都需要熟悉。面试题有两类，一类是考察你对一门技术的深入程度，另一类是考察你的技术广度。面试官考察你的技术广度的时候，可能会提及多个技术方向以及相关的名词，其中你有不会的或者不熟悉的，是很正常的事情。不要因为回答了几个“不会”就暗自得出一个结论：“完了，我肯定没有希望了。”，其实不是这样，面试官会综合考虑的。例如，我在面试测试工程师的时候，我会考察他（她）的测试能力、开发能力、数据库、外语、对一些特定产品的熟悉程度等多方面，最后得出一个综合的评价。一旦你给自己一个不能通过面试的暗示后，你的心就会发慌，你在面试中的表现就会越来越差，这个时候，你离期望的职位就真是越来越远了。

不要夸夸其谈。如果面试官的问题正好是你熟悉的，这是好事，但是，一定要避免夸夸其谈。在我们的传统文化中，不喜欢不谦虚的人。另外，一旦你夸夸其谈，你可能就变得浮躁，容易说一些错误的表述，而一旦出现技术上的错误，面试官对它会很敏感，它会成为你的硬伤。

如果遇到一些很特别的问题，不要惊慌。我的一个朋友告诉我，微软公司曾经对她做过一次面试，其中有一道题：“你如何测试一个杯子。”。这个问题在现实工作中是不存在的，因为大家应聘的是软件测试工程师，而不是做杯子的技师。这个问题很特别，你难免会有些惊讶，换了是我也会这样。我想告诉你的是，既然这个问题在实际测试工作中不会出现，为什么微软公司仍然会把它列为技术面试题呢？其实，面试官只是想考察你的应变能力和发散性思维，想通过这个问题看看你的思维过程和思维能力，

面试官也不会有一个统一的答案。你只需要陈述你的想法即可，想到什么就说什么。如果你一听这个问题就不知所措或者直接说“不会”，面试官就会给你定下一个否定的看法。

以上都是一些锦上添花的技巧，真正决定你在技术面试中的通过率的是你的技术实力，这是你能否找到一份技术工作的本源。

3.8.4 情景问答

面试，对于应聘者是一个考验，而对于面试官来说又何尝不是呢？他们需要从众多的应聘者当中发现合适的人选，并且日后这些新人的工作业绩多少都和面试官有点关系。例如，如果新人到单位上班后，表现不佳，大家可能就会嘀咕：“这是谁招进来的？”。为了了解应聘者技术以外的处事态度、沟通能力等等，有的时候面试官会用情景问答的方式来考验应聘者。

所谓情景问答，就是描述一个场景，并分配应聘者一个角色，让应聘者处理当中的问题。有的时候还需要应聘者以这个角色的口气说话或者行动，以便更加贴近真实情况。举个例子，在实际工作中测试工程师和开发工程师有可能发生矛盾，如果测试工程师没有理性地处理矛盾的办法，往往会把事情弄得更糟，所以这也是招聘单位比较担心的一个方面。有的单位在招聘测试工程师的时候就有这方面的情景问答，如下面的例子：

“假如你是一个测试团队的工程师，项目已经到了后期。这个时候你发现了一个比较严重的bug，因为很严重，所以你及时做了提交。在当天的测试和开发人员的联席会议上，一位相关的开发人员发言，说到了这个bug，认为这是你的失职，因为这个bug应当更早就被发现，现在发现已经有点晚了，让开发人员有点被动。在这样一个场景下，你做何反应？你会如何做解释或者反驳呢？”。如同前面我提到的，有的时候面试官会要求你进行角色扮演，以“这位测试工程师”的口气来解决问题。

也许有些朋友会认为，这没什么难的吧？即使要求扮演，但毕竟不是在实际工作中，可能不会出现那种火药味，肯定不会“吵起来”，所以大家都会显得有耐心，这不就迎合了面试官的预期了吗？面试官当然也会考虑到这一点，为了真实测试出应聘者的态度，他不会满意于你平静的解释，他会继续模拟“那位开发人员”来挤兑你、指责你，加上你毕竟是在被面试中，所以还是有些压力的。

即使“那位开发人员”说的话再刻薄，但毕竟是面试官模拟的，就是脾气再差的人也不会和面试官吵。但是，应聘者说出来的话和表情却是各不相同。遇到这种情景问答时，我们需要注意些什么呢？

首先，是要平静。因为是在面试，应聘者多少会感觉到一些压力。心理压力大，你在这种情景问答中就容易慌张，导致表现不佳；压力小，你就会从容一些。在心理压力方面，你是可以自己调节的。对于一家公司提供的工作机会，你既要有一种想得到它的积极性，又要有一种不行就算了的大度，这种“若即若离”的态度会有利于减轻心理压力。

其次，带着表情去说话。如果你毫无表情，即使说得有道理，面试官可能还会有点疑问：“说得是不错，但是怎么像说别人的事？一点表情都没有？”。言词恳切最能打动人，这里面表情是很有作用的。例如，如果你要表现出你没有被激怒，而是在对开发人员做一个解释，你可以面带微笑地去说，我相信你回答得会比较好。

再次，应当知道一点，情景问答没有惟一的答案，只要你能自圆其说就可以。例如，有的人就说，我会简单解释一下，如果开发人员还是追着说，我就不会再说什么，因为这样容易吵起来，我会在会后再单独找他；有的人则会把一些客观原因说一下；另有一些人则会道歉，责备自己。在我看来，这些回答都是可以接受的，因为都没有激化矛盾。

从表面上看，情景问答对人的机灵性有点要求，其实这只是表象。如果你在日常工作中真的是一位耐心、和心静气的人，在面试中会自然表现

出来的。

3.8.5 人事面试

现在很多公司在技术面试之外，还有人事面试，就是由人事专员对候选人做面试。这个环节因为没有具体技术问题，相对来说“虚”一点。有的朋友会说：“容易！”，觉得没有难度，认为只是一个过场而已。更有甚者，对人事面试非常不屑，心里想：“技术上什么都不懂的一个小女孩，凭什么面试我？”，不把人事面试当回事，在人事面试当中表现得不诚恳，这就“大意失荆州”了。

术业各有专攻，计算机是一个行业，人事也是一个行业，论资格来说，人事这个行业可比计算机资格老得多☺。你拿你自己的计算机技术和人事专员比，难免有点难为人家。既然很多公司都有人事面试这个环节，说明这个环节还是比较重要的。人事专员在是否录用你的决定中是很有作用的。的确，通过人事面试的候选人不一定被录取，因为还要看他（她）的技术能力。但是，人事面试没有通过的是一定不会被录用的。你想，在有多个不错的候选人的时候，如果人事专员对候选人A的评价是浮躁、合作意识差、傲气等，A怎么会被录用呢？

那么，在人事面试中，面试官（人事专员）会考察什么？

人事专员的主要任务是，从候选人的穿着、言行举止中去感觉候选人的性格、工作态度、合作能力、稳定性等等。相对来说，在技术面试中，面试官不大在意候选人的穿着、坐姿、语气等，因为焦点在技术上；而在人事面试中，这些就变成了主要的考察点。我们来分析一下人事面试中被关注的几个方面：

首先，你的外表。在面试中，最合适的穿着应当是要端庄。端庄的外表容易给人留下认真、在意这个工作机会、尊重本公司的良好印象。我相信大家不会穿着奇装异服去面试，不会去冒这个险，但同时其他几点也是

需要注意的，例如衣服不要太脏，不要破，钮扣打开不要过低（男性也要注意这一点），头发不要蓬乱，不要满头大汗等等。另外，人也应当比较干净，我记得有一次见到一位候选人，几个手指甲里全是黑泥，让人“敬畏”。

接着，你的言行举止。有的时候，你发现人事专员的问题特别简单，无论谁都会回答。那你有没有想过，这么简单的问题人事专员为什么还要问呢？她们并不是在浪费时间，对她们来说，有的时候更看重你说话的姿态、语气、手势等等，你所说的内容有的时候倒被略过了。我不止一次得到人事专员给我关于候选人的负面反馈，说有的候选人说话的时候小动作多，不专心；或者一边说话一边晃动，太随意。这一方面，只要你重视人事面试，潜意识对身体就会有所约束，给对方留下的印象自然就好得多。

最后，人事专员也的确会问到一些问题，例如，你为什么离开上一家公司？你对自己的职业发展有什么考虑？你希望到一个什么样的公司工作？除了申请的这个职位之外，你还有没有可以考虑的职位？你以前的薪金是多少，这次的期望呢？为什么会在薪金上有这个期望？等等。你可以事先准备一下，因为有些问题的确需要回答得委婉一些，或者避免面试官的误解。

从整体上看，在人事面试中，最重要的是要看重它，而不要把它当作笔试和技术面试之间的休息。

3.8.6 抓住主动的机会

客观上来说，候选人在面试中（包括笔试）是被动的。笔试的题是之前出好的，面试中的问题是面试官想好的，候选人都只能被动地接受。可能大家也想到过，作为候选人，我们有没有可能在面试中争取一些主动的机会呢？如果能够这样的话，会有利于我们找到合适的工作。

虽然我们处处都在被人考察、被人问，但是，不要忘记，至少我们的

内心是可以主动的。例如，候选人进入公司后，可以观察这个公司的办公环境、氛围；从笔试题中大致猜测出题人的技术层次和公司招聘的技术方向；在面试过程中，可以稍许地分一点心来观察面试官。一般来说，虽然面试官不一定是这家公司的高级管理人员或者技术尖子，但至少是很受认可的。你可以把面试官当作一面重要的镜子，来看出一家公司的层次（所以，公司在决定面试官人选的时候也要注意啊☺）。面试官所问的问题、姿势、语气以及神态，都可以去多注意一下，而不是只局限在回答问题上。找工作的事情毕竟是一个双向选择的过程，如果你从自己的观察中感觉到这家公司不是自己所期望的，当录取通知来临要做决定的时候，就要细思量了。

在内心暗自观察是一种主动，有心之人都可以为之，除此之外，还有什么主动的机会我们可以把握吗？

在面试的尾声，无论是原定的安排还是出于客套，现在很多面试官都会说："我想要问的问题就这些了，你看你是否有问题想要问我或者想要了解的？"，这是个好机会，你要把握住。如果你心里没有准备，当这个机会来临时，你不知怎么作答，几秒钟的时间，面试官看你没有问题，就会正式宣布面试结束，礼貌地送你出去。这可能是面试过程中候选人惟一一次可以自由发问的机会，如果你想把握住它，你可能需要预先准备一下。

你不要认为问问题是一种不恭的举动。既然面试官给了机会，哪怕是客套，顺着这个话题问一些问题是不会有什么不恭的嫌疑。虽然在现实情况中大多是公司在挑选职员，公司占主动和支配地位，但从本质上说，二者是平等的，候选人可以发问。

问题如果问得比较合适，既能让自己增进对这家公司的了解，还能让面试官感觉到你很积极，在认真地对待这次面试。例如，下面这些问题可能是很多候选人都想知道答案的：

（1）公司的主营业务和发展方向；

（2）公司规模和结构；

（3）本职位所在的项目及其进展；

（4）这个职位期望招到什么样的人？

还有一类问题也是候选人可以考虑问的，当然，你不一定能得到答复。我就遇到过几次候选人问我："刚才在面试（或者笔试）中有一道题可能答得不好，你是否方便告诉我正确的思路是什么？"。之所以说你不一定会得到答案，是因为面试官可能会出于保密的需要而不回答你，但问一问没有什么，是可以的。从我个人来说，我会认为这样的候选人比较积极。

如果心里没有准备的话，就不要问了。因为毕竟你还是在被面试，如果问了一个唐突的问题，的确会有一些负作用。例如，即使你真的因为各种原因（例如面试的公司太多）不知道这家公司的名称，也不要去问："请问你这家公司的名称是什么？"。这就好像和一个朋友聊天了好一会，最后问一句："不好意思，你叫什么？"，虽然诚实，但的确让对方很没面子，感到惊讶。

如果面试结束了，面试官并没有主动给你提问的机会，你看他已经在收拾面试记录准备走了，你也可以主动发问："不好意思，可不可以耽误你1分钟的时间？我对公司很感兴趣，所以想问几个问题。"，我相信一般你都不会被拒绝的。

你还可以借着主动提问的机会尝试着去取得人事专员的联系方式，这样，你就可以主动联系这家公司了。即使没有被录取，你还是可以通过主动联系得到没有被录用的原因，而不是石沉大海。失败的原因对于找工作的人来说是宝贵的信息，对你是很有帮助的。如果你能建立起与人事专员的联系，那就更好了，多一个朋友多一条路，如果以后这家公司还有机会的话，你会优先得到相关信息。

3.8.7 外包职位的面试

随着我们国家软件业的发展，一种新的职业形式越来越被人们所知——外包。外包有多种形式，我们这里专指On Site形式，就是员工的工作地点在客户公司的内部。Off Site形式的外包与非外包职位没有什么不同，所以不作讨论。因为可能有的朋友对外包还不大了解，我这里举一个例子来说明。例如，公司A与微软公司签订了外包服务协定，为微软提供外包人员。你到公司A去应聘，通过了这家公司的面试后，他又把你推荐到了微软公司去面试。在你通过微软公司的面试后，你就得到了一个外包职位。你与公司A签订劳动合同，是公司A的员工，但日常的上班都在微软公司。具体结算方式是微软与公司A结算，公司A给你支付工资。外包职位的好处是它比直接成为外企员工的门槛低一些，而你一样在外企内部工作，是一个不错的学习机会。

为叙述方便，下面把类似公司A这样为大企业提供外包人员的叫做外包公司，把微软这样接受外包人员的叫做客户公司。

可能大家也看出来了，外包职位的应聘过程多了一轮面试——客户公司（例如微软）的面试。那么，这一轮面试有什么特点呢？

外包职位与非外包的普通职位有一个比较大的不同，那就是外包职位更喜欢一个通才。拿软件测试外包职位来说，希望你有经验，懂测试，有比较好的计算机专业知识，人积极，良好的团队合作精神，英语好等等，下面我分别说明一下：

1. 相关的工作经验

就一家客户公司来说，因为他有多家外包公司提供人员，他对人员的期望就会比较高。因为他的选择余地很大，一个职位可能有超过10个条件不错的应聘者在竞争，所以没有相关工作经验的候选人当然不会有竞争力。另外，外包职位一般都是做软件产品，对人员的要求相对高一点。

作为通例，客户公司的面试一般也会从你的工作经验谈起，然后慢慢

扩展开。

2. 比较好的计算机专业知识

还是拿软件测试的外包职位来说，因为我对这一块熟悉一些，客户公司希望候选的测试工程师要对以下几方面都比较熟悉：

（1）Windows操作系统，以及其他非Windows操作系统；

（2）数据库访问；

（3）编程语言，最好熟悉一种编程语言；

（4）软件工程知识，以及软件测试方面的知识。

客户公司会有意识地在面试中提及以上各方面的内容。一般来说，客户公司会有多个项目，他们希望候选人有比较好的基础，是一个通才，这样候选人适合做的项目会多一些，他们也好安排具体工作。

3. 积极的态度

这也是一个基本的要求。我相信即使不是外包职位，面试官对这一点也是很看重的。一个人积极不积极，在项目中起到的作用完全是两重天。现在谁还愿意像管小学生一样管理员工呢？公司都希望员工有积极的心态，专注地投入到工作当中。在客户公司中更是这样，因为客户公司都很大，人力成本和管理成本很高，项目的时间表又会比较紧，他们不会要一个态度不积极的人。所以，在参加客户公司面试的时候，候选人的精神风貌一定要好，表现得要积极。如果你的问题都回答对了，却给面试官留下一个不积极的印象，通过的概率会很小。

4. 良好的团队合作精神

客户公司的项目一般都比较大，而且都是很正式的软件产品项目，一个项目组动辄几十个人，在这样的大团队中，良好的合作精神也是客户公司所看重的。不然的话，即使一个人技术再强，如果他不合群，不能与他人进行良好的合作，你说，客户公司如果把这样一位“大侠”招进来，不

是给自己的项目找麻烦吗？在面试当中，面试官直接问你“你是否注重团队合作？”的可能性不大，因为答案肯定都是一样的。面试官会从你的言行举止和你以前的工作经历中去揣测你的团队合作精神。

5. 英语好

客户公司的项目一般都是国际化的项目，工作语言为英语，所以面试中会有对英语的考察。英语阅读和书写的能力排在第一位，例如，拿一段英文，让你在几分钟内翻译成中文，或者把一段中文翻译为英文。有的职位对口语有要求，但也只会是基本的要求，例如自我介绍，介绍一个自己做过的项目等等。

在面试形式上，除了“面试”以外，有的客户公司会有笔试和电话面试。这两种方式都是为了甄别候选人是否达到了面试的标准，以减少面试量。电话面试就是双方约定一个通话时间（也可能没有预约），面试官给你打电话，和你简单沟通一下。如果你遇到正好不方便接电话的时候，一定要直接和面试官讲清楚，再另约时间进行电话面试，否则你这边因为不方便，讲话支支吾吾，而面试官不知道，很影响你的成绩。

3.9 有作用的一环——记录结果并做分析

投了简历，做了笔试、面试，可能你已经有些疲惫，之后还需要做什么吗？

找工作的过程是纷杂的，我们在集中精力寻找机会、投送简历，心情忐忑地等待面试通知，赶场似的奔走于各个公司之间以应付面试，翘首期盼公司的录用通知……等等这些，我们在找工作的过程中要重复多次，要想在这个纷杂的过程中坚持做一些记录并不容易。有的时候只能在网吧上网，不方便记录；有的时候连续地投了多份简历，懒得去记录；有的时候是在拥挤的招聘会上，不知道怎么记录；有的时候是朋友介绍的，根本就

没有想到记录……

但是，毫无疑问，在找工作的过程中多做记录是有用的。要记录都给哪些公司发送了简历？得到了哪些公司的面试机会？在面试中有哪些回答和举动自己认为做得还不错的，而又有哪些是不足的？得到了公司的哪些反馈？你可能会问，如果我把这些记录下来，挺麻烦的，而我能从中得到什么？

除了能让你日后翻翻怀旧之外☺，通过做记录和后期的分析我们可以实现：

（1）通过对记录的分析我们可以知道自己在哪一方面有竞争优势。通常你可能会同时应聘多种职位，例如你希望做一个软件测试团队的管理者，但高级测试工程师的职位也在考虑之中。这时你可以把自己得到的面试机会都记录下来，并做一个简单分析：如果得到管理职位的面试机会多一些，说明你在这方面比较有竞争力；反之，如果是测试工程师方面的面试机会多一些，则说明可能你在管理方面的才能还没有得到认同，你可以考虑是不是需要调整求职方向或者修改简历。作为一个求职者，很多时候我们只能去尝试，通过尝试逐步揣摩社会对自己的评价。如果在找工作的过程中多做一些记录，会缩短这种揣摩的过程，及时看清自己的优势和不足。

（2）通过对记录的分析我们可以实现自我提高。笔试、面试都是我们自己单独参与，没有朋友能提出什么具体的建议和意见，对笔试、面试过程的记录，就是给自己一个回顾和反省的机会，从中汲取经验和教训，会让自己在下次的笔试和面试中表现得更好。找工作的过程其实是有成本的，如果缺少了这种回顾和反省，一而再、再而三地碰壁，求职的成本就会很高。

（3）用人单位的反馈是宝贵的，请记录下来。有的时候，你最终没有被某家公司录用，但你可能会得到反馈：“很抱歉，先生（女士），我们觉得你挺不错的，但是……，希望下次能够合作。”。这种反馈虽然没有

录用通知来得让人兴奋，但也是很有用的，它有利于你再一次反省自己，提高自己。一般来说，只有在面试后你主动联系公司时，才有可能得到这样的反馈，不然，现在绝大多数的公司都是委婉地用“杳无音讯”来告诉你没有被录用。

在找工作的过程中做些记录，避免了找工作的过程出现一本胡涂帐。如果找了一段时间的工作，工作还是没有找到，对自己的认识又没有得到提高，那岂不是白费功夫？

任何持续的事情都是有难度的。就好像戒烟一天容易，坚持戒烟很难；锻炼几天容易，坚持锻炼很难。对找工作过程的记录也是这样，在找工作的过程中做几次记录是容易的，坚持做记录和分析就难了。如果你能坚持，肯定会有收获。

3.10 不要气馁

因为一些客观的原因，求职者的人数大大多于职位数，所以现在要找一份工作是件不容易的事情，而对于刚毕业的朋友，或者想转行的朋友，就更显艰难。如果你投了N份简历，得到的面试机会却很少；如果你面试了N次，录取通知却仿佛在和你玩捉秘藏，始终可望而不可及，这时或许你会有些伤感，甚至怀疑自己，自己究竟行不行？

我理解你，朋友，在口袋里的钱消耗得差不多的时候，在被人问起“在哪儿工作？”而不知如何回答的时候，在不知何去何从的时候，内心中感到有些气馁，这很正常。在这个时候，我想对你说一些话。

面试（包括笔试部分）是比较客观的，但也有主观性。对于笔试来说，如果考察的知识点是你熟悉的，你很容易通过，反之，如果比较生疏，分数可能就低了。面试的灵活度更大一些，如果你和面试官气质相似，你很容易获得对方的青睐，对方也容易原谅你并不完善的回答，反之则容易被

挑刺。所以，面试实际是用人单位用他们的“尺子”来衡量你的过程，符合他们要求的就通过，不符合的没有机会。没通过，并不表示你不行，只能说你与这把“尺子”不相吻合。如果你十次面试都没有通过，也只能说明你与十把“尺子”都不吻合。这个世界上的公司千千万，“尺子”也就千千万，说不定机会就在下一次。

偶尔的伤感或者气馁是很正常的，只是不要让它长期占据了你的心灵。每一次失败后，我们要去总结，去分析，然后再学习，再提高，保持积极的状态，时刻准备着下一次的尝试。找工作，就像是一个人跑的接力赛，不管前几棒跑得如何，我们应当始终关注：下一棒马上就要开始，我们准备好了吗？

不要气馁，朋友，坚持就是胜利！软件测试的大门始终向你开放。

04

第4章 软件测试技术基础

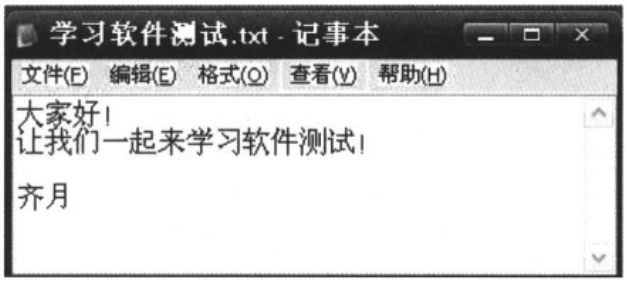
如果你想找一份软件测试的工作，而对软件测试一点都不懂，那是肯定不行的，不要心存侥幸。如果对软件测试你还是个门外汉，你可以通过阅读本章来迅速地学习一下这个领域的技术基础；如果你曾经学习或者接触过测试，本章内容则可以作为复习的资料，用以把自己的知识恢复到最好的状态，去迎接面试。本章介绍的测试技术基础，我敢保证不会枯燥，不能供你做催眠之用☺。

4.1 功能测试

对于软件测试初学者来说，专业名词的学习是必要的，同时也是容易“陷”进去的地方，被一些难以区分的术语弄得一头雾水。随便找一篇测试方面的文章，你很容易就看到白盒测试、黑盒测试、功能测试、性能测试、压力测试、可用性测试、界面测试、兼容性测试等名词。为了尽快了解软件测试的实质，我建议大家不必在这些名词上绕圈子，接触测试的时间长了，有些东西必然会明白。现在我就从一个核心概念说起，让大家对软件测试都需要做什么有一个初步的了解。就我自己来说，我是把“功能测试”排在第一位的。

要做一个软件，有三种角色会参与进来。第一种角色是定义需求的，我们称之为程序经理，他们定义软件要做成什么样，他们来写出软件需求规格说明书；第二种角色是开发人员，就是挥汗如雨写程序的，他们把软件编制出来，所依据的当然是需求规格说明书；第三种角色就是测试工程师，测试工程师戴着眼镜，一手拿着需求规格说明书，一手拿着软件，一一比对，专门找问题☺。那么，功能测试怎么做呢？下面我举例来说明。

Windows操作系统附带的记事本相信大家都用过，如下图所示，这个软件很小巧。下面通过记事本软件的测试来说明功能测试如何做。



下面我给出一个假想的需求规格说明书：

明日之星软件公司记事本软件需求规格说明书

我们要编制一个字处理软件，它应当实现以下功能：

1. 文件操作

（1）用户可以新建文件；

（2）用户可以打开一个已经存在的文件；

（3）用户可以保存编辑的结果到硬盘上；

（4）用户可以把编辑的结果另存为另外一个文件；

（5）用户可以打印编辑的结果，而且，用户在打印前还可以进行页面设置；

（6）用户随时可以退出编辑。

2. 文字编辑

…

3. 格式设置

…

…

作为一位测试工程师，对软件做功能测试，就是按照需求规格说明书的要求，一一检查软件产品，看软件产品是否正确实现了应有的功能。在上面的例子中，我们就要去验证，通过这个软件用户是否可以去新建文件，是否可以打开文件，是否可以保存文件，等等。这些就是功能测试。

听了我对功能测试的解释，你是否会因此而认为功能测试很简单？哈，不要高兴得太早。做功能测试的难点之一就是，你如何能下结论说一个功能是正确的？例如，把编辑结果保存为文件这个功能，为了验证它是否正确，我们需要考虑很多种情况：

（1）各种不同的存储路径；

（2）各种不同的文件名格式；

（3）不同的文件的长度；

（4）操作系统的不同；

（5）硬盘分区的不同情况；

（6）是否正确保存了？

（7）快捷键；

（8）硬盘空间不够的情况下如何处理？

（9）语言环境的不同；

（10）如果已经打开一个“保存”对话框，还能点击“保存”命令吗？

（11）如果要保存为文件A，而文件A正被另外一个程序锁定，这时怎么处理？

（12）如果...

要测试各种的可能性似乎是无穷尽的，我们要想得尽可能地多，然后通过测试来给出我们的结果：这个功能是正确的还是并非如此。所以，你可能已经看到了，绞尽脑汁是测试工程师的平常状态，这也是测试的难点，当然也是测试工程师的特长。

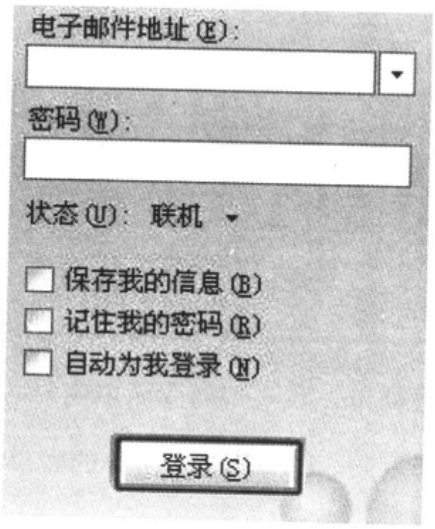
另外，还有一点，我们还要在软件产品里找一找是否存在需求规格说明书里没有的功能。可能大家觉得这是多此一举，谁还会吃饱了撑的做这种事？这都是不好说的，有可能开发人员因为误解了需求规格说明书或者单纯因为私人偏好，实现了一个或者几个不应当存在的功能。哪怕这个功能再酷，也是错误的，测试工程师应当发现并作为问题提交上去。

说了这么多，我们来概括一下，功能测试要做的是把需求规格说明书中的功能定义与软件产品一一比对，力争发现其中的错误。功能测试难吗？不难，是有章可循的，需求规格说明书就是我们的“红宝书”。功能测试容易吗？不容易，要考虑周到很不容易。

4.2 软件测试思想

4.2.1 等价类的划分

软件测试该如何做？如果你是诚心想在这一方面发展，我相信你不会认为软件测试很简单。如果真的是无论谁都能做好，这个职业就不会有什么竞争力，也不会有什么吸引力了。在一款软件面前，随便地敲打键盘输入一些字符，用鼠标点击几下，看看能不能用，我们不能将其称之为“测试”，勉强也只能说是“试用”，因为这很容易挂一漏万。例如，大家都比较熟悉的MSN登录界面，如下图所示：



MSN登录界面一共有电子邮件地址、密码、状态、保存我的信息、记住我的密码、自动为我登录和登录按钮7项需要考虑。简单化一点，就说电子邮件地址和密码两项，其组合何止千万种情况，如果仅仅是随便想几种情况进行测试，微软公司怕是信不过你的测试报告的☺。测试的目的就是要尽可能地发现bug，而这只能通过严密的测试思维来达到。

我们来举一个生活中的例子。你和好友们一同到饭馆去大吃一顿，在点菜的时候，你一般会这样考虑：来几个荤菜，几个素菜，一两个凉菜，再来一个汤，菜就点好了；酒水方面，来几瓶啤酒；主食方面，简单的话

就每人一碗米饭，这样就全点好了。在菜单上，菜、汤、酒、主食都有很多种，我们只会选择其中的几种有代表性的（就是我们很想吃的），我们认为这样就可以了。这个举动里面有一种思想对我们测试很有用，那就是分类。

MSN登录界面的那个例子我不接着讨论了，如果你感兴趣，你可以自己琢磨一下。我下面列举另外一个例子：计算器的加法测试，计算器的界面如下图所示。

这个计算器软件很简单，我再把问题简单化一些，讨论一下加法如何测试。能简单就不选择复杂，我想这是一个好的做事方法。

虽然这个软件很简单，但是加数的选择范围可以说是无穷尽的，我们不可能列举完。所以，我们要对这些加数进行分类，然后从不同的类别中选择有代表性的就可以了，就好像我们点菜一样。现在的问题是，如何分类呢？

第一种分类方法：加数可以是正整数，正小数，零，负整数，负小数。两个加数，各有5种选择，排列一下就有了25种情况。

第二种分类方法：分类的标准是有无进位，这样我们至少有两种情况需要测试：结果（就是“和”）有进位，结果无进位。

第三种分类方法：按照加数的位数来分，例如加数可以为一位数，两

位数，三位数，四位数，……。如果这个计算器支持32位数之内的运算，我们可知加数有32种选择。在这种情况下，我们没有必要把这32种选择都尝试一遍，在超过10位数之上的，我们可以每5种为一类，如果其中一个通过测试，我们可以认为其他的也没有问题。例如，15位数到20位数，我们可以选择18位数进行测试。

第四种分类方法：以加数的个数来分，2个，3个，4个等。

第五种分类方法：……。

……

我们再回过头来把上面的例子总结一下，在考虑一个功能的测试的时候，我们可以做一个分类，从每一类中选择有代表性的数据或者情况来做测试，这在术语上就叫做等价类的划分。在这个重要的知识点上，首先是我们要知道这个概念，接下来，也是最重要的，根据实际情况去做合适的分类和选择有代表性的数据去做测试。

4.2.2 边界值

边界值的测试是一种比较重要的软件测试思想，也很简单，简单到几句话就可以说完。

我们还是先从实例入手，例如，假设你要测试一款税务软件中计算个人所得税的功能，假定根据国家的规定，个税从1600元起征，如果你来测试，你要做怎样的考虑呢？在这个例子中，1600是一个重要的数值，它就好像是两个国家的边界，边界两边是两重天。就算是没有接触过边界值测试理论的朋友，1600这个数值也会在他们的测试考虑之中。那么，是否懂得这种理论的不同之处在什么地方？边界值的测试思想告诉我们，在确定了一个边界值以后，还应当在边界值的左右两方各选择一个数值进行测试。在本例中，就1600这个边界值来说，我们应当考虑的是1599、1600和1601。经过了三次测试以后，我们可以大致保证计算个税的功能在1600这个边界

值的实现是正确的。

为什么我们需要做边界值的测试呢？任何的要求一般都会有其原因所在，我们可以尝试着讨论一下。边界值是程序员容易出问题的地方，所以测试员需要重点关注。当然，如果程序员只需要处理一个类似“1600”的问题，那出现错误的概率是很小的，除非他写程序的时候睡着了☺。但是，现实的软件总比我们的例子要复杂很多倍，当程序员需要处理100个类似“1600”这样的边界问题时，他的脑子里可能全是“大于”、“大于等于”、“小于”、“小于等于”等条件，如果再加上一边写程序一边与朋友通过MSN聊天，或者因为某事中断了程序的编写而在以后继续编写的情况，出现几个错误也就难免了。

边界值理论虽然简单，但在实际的运用过程中也容易出现问题。我简单说一下这些问题，你可以做为一个参照。

第一，分析问题时只懂得边界值理论，再没有其他的法宝了。要完成一个功能的测试，通常都是数种方法的综合，如果你只会这一“板斧”，软件质量是很难得到保证的。例如，如果我们把等价类的划分与边界值综合起来用，会取得更好的效果。

第二，在问题复杂的情况下，往往会遗漏对一些边界值的检查。如果只有一两个功能需要测试，其中的边界值不过五六个的话，谁都知道要怎么测试。但是，这么简单的任务在现实工作中是微乎其微。作为一位测试工程师，我们要面对的可能是几百个边界值，我们是否对每一个边界值都做了测试，就要看我们思维的严谨程度了。

第三，有些边界值看起来是不重要的，或者出现概率很低的，你容易“放它一马”。例如，假设MSN最多允许用户拥有1000个联系人，这个边界值你愿意测试吗？你可能会想，不可能吧，1000个联系人？200个撑死了。所以就放过1000，转而去测试200。这种以自己的观点去替换边界值的做法很危险。再如，在计算器软件的测试中，你是否真的会把最大数（32

个9，如下图所示）纳入你的考虑范围？你会去检查这个数再加1的结果是多少吗？

道理总是简单的，而要坚持运用却不容易，或许很多道理都是这样吧。

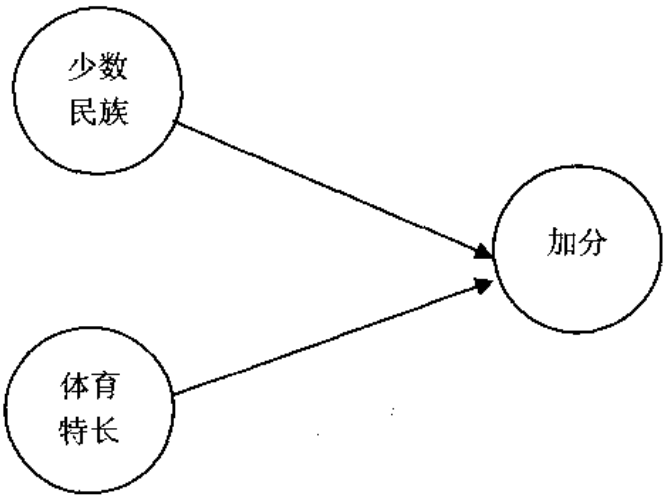
4.2.3 因果图和判定表

1. 因果图

因果图也是软件测试中经常涉及到的基本方法之一，从名字上我们可能看出，似乎是原因和结果相关的，我们通过一个实例来看一看它究竟是什么意思。

例如，在一款招生管理系统中，有一个计算加分的小功能。加分的政策是：“如果考生属于少数民族，加5分；如果考生是体育特长生，加5分；如果考生具备了前两项加分条件，为了保持公正，加8分（而不是10分）。”。

这项政策在软件方面就是需求定义，它定义了加分的条件（原因）和具体加分额度（结果），我们可以用图的形式把它表现出来：



上图就是因果图。我们通过文字形式对需求做分析，把其中的原因和结果通过图的形式表现，让人一目了然。做因果图的目的是为了把软件产品的内在逻辑整理清楚，方便我们做测试。即使是非常聪明的人，在条件一复杂的情况下也容易出现遗漏或者差错，通过因果图则能有效地避免这样的问题。

2. 判定表

判定表则比因果图更清晰，在实际测试工作中判定表应用得要更广泛一些。我们用判定表的形式把上例的需求做一个分析：

需要测试的情况	情况1	情况2	情况3	情况4
少数民族	是	是	不是	不是
体育特长	是	不是	是	不是
加5分		加	加	
加8分	加			

大家在看上面的判定表时，可以按照列来看，各列中的“情况1”、“情况2”、“情况3”、“情况4”就是我们需要测试的不同场景，这对我们编

写测试用例有指导意义。如果需要对判定表做一个大概的描述的话，可以这么来理解：我们把各种原因和结果列在一张二维的表格中，严格地表现出它们之间的逻辑关系。

上面给出的例子是很简单的，因为我认为简单的例子更有利于说明问题，实际工作中的因果图和判定表则要复杂得多。例如，如果再增加一个加分条件，判定表就会变成下表：

需要测试的情况	情况1	情况2	情况3	情况4	情况5	情况6	情况7	情况8
少数民族	是	是	是	是	不是	不是	不是	不是
体育特长	是	是	不是	不是	是	是	不是	不是
照顾条件	是	不是	是	不是	是	不是	是	不是
加5分								
加8分								
加N分或者其他优惠								

大家可以在学习软件测试的过程中或者实际测试工作中尝试使用因果图和判定表，使我们的测试思维更加严密。

4.2.4 代码覆盖

大家都知道，软件是由一行一行的代码垒起来的，这样自然就会想到，如果我们能够尽可能多地测试到这些构成产品的代码，软件产品的质量就会更有保障，这种想法是对的。所以，在一些大的软件研发项目中，会以测试达到的代码覆盖率来作为衡量软件测试工作的指标之一。我下面举例说明3种常见的代码覆盖。

1. 语句覆盖

所谓语句覆盖，就是说测试人员要通过测试用例把代码中所有可执行的代码都执行过，例如：

```
…
if （a==0）b=-1;
…
```

在这个代码片段中，“b=-1”是可执行的语句，我们在做语句覆盖的时候，就要保证这个if语句的判断条件为真，即“a==0”，这样“b=-1”就会被执行。

2. 分支覆盖

我们来认识一下分支覆盖，先来看一段代码：

```
…
if （（a==0）&&（b==0））
{
        c=1;
}
else
{
        c=2;
}
…
```

所谓分支覆盖，就是要通过测试用例把分支语句中的不同分支都要测试到，在上面的这个例子中，要测试到条件为真和假两种情况，即要确保“c=1”和“c=2”两种情况都得到测试。至于如何达到判定条件“（（a==0）&&（b==0））”为真或为假，分支覆盖没有强制要求。

3. 条件覆盖

条件覆盖比分支覆盖要求更为严格，它的要求是：对于每一个在分支判断中出现的条件，我们都要测试它为真和为假的情况。例如，在“if （（a==0）&& （b==0））”中，我们既要设计测试用例去测试“a==0”和“a!=0”的情况，也要去测试“b==0”和“b!=0”的情况。同时，条件覆盖不要求对a和b的取值情况进行排列组合，即选取两个用例就可以达到这种效果，例如a==0 且 b==0，a!=0 且b!=0（还有其他用例也可以达到这种效果）。

事实上还存在多种覆盖，但都是基于语句覆盖、分支覆盖和条件覆盖的，只要你把这几种覆盖的概念掌握了，后续的理解就不会很难。因为代码覆盖比较复杂，我这里仅做一个简单的说明，大家如果还想深入了解，请参考相关书籍。

4.3 如何编写测试用例

在前面我们谈了软件测试的一些基本概念，例如什么是软件测试，软件测试的基本要求，什么是功能测试，软件测试的4个基本思想：等价类的划分、边界值、因果图和判定表、代码覆盖。现在似乎到了我们要动手试试的时候了，如果你一听到要动手做测试，就捋起袖子敲键盘点鼠标，那就太急了☺。做软件测试有一个重要的步骤——编写软件测试用例。

什么是测试用例呢？测试用例其实就是一个个你测试的想法，你有了这些想法以后，详细地写下来，就成了测试用例。测试用例有几个重要的组成部分：

（1）简明扼要的标题；

（2）详细的步骤；

（3）正确的预期结果。

我们还是通过一个例子来说明。例如我们在测试记事本的时候，有了一个想法：应当测试一下这个软件能不能编辑中英文混合输入的内容，如下图所示。为了准确地实现我们想要测试的思想，我们要把它写下来，并且写下的内容要让任何人来看都没有歧义。

测试用例：验证记事本程序可以编辑中英文混合的内容

测试步骤：

1. 运行记事本程序；
2. 切换到中文输入法，输入中文“学习编写”；
3. 切换到英文输入法，输入英文TestCase；
4. 保存文件，文件名为testcase.txt；
5. 关闭记事本程序；
6. 双击testcase.txt以打开文件。

预期结果：

1. 文件的内容是“学习编写TestCase”，如下图所示。

测试用例还有一个优先级的概念，就是用来区分哪些用例更重要。一般可以分为5个级别，分别用0~4来表示，数字越小表示越重要。如果项目小，优先级的好处不容易显现出来。当项目比较大，时间又不宽裕时，可能只能执行更重要的测试用例，这个时候优先级的重要性就体现出来了。

大家也看到了，其实写测实用例并不难，但是它仍然容易出一些问题，例如：

（1）含混不清或者与内容不相符的标题。例如，上面的例子，如果用例叫“验证记事本可以编辑内容”，这个标题就没有准确表达出测试用例的实际内容。

（2）过于简单的步骤。这是一个容易犯的错误，很多朋友在编写用例的时候，总是写得很简单，例如上例中的多个步骤可能就会变成惟一的一步：“输入‘学习编写TestCase’”，如果不是作者本人，其他人来看，肯定会引起歧义，怎么输入，是用键盘还是用拷贝的方法？那么写测试用例要详细到什么程度？就是让一个不了解你的工作的人来看，如果他的理解和你一样，说明你已经表达清楚了。

（3）没有写明预期结果。这是个严重的问题，如果没有预期的结果，那什么是对的什么又是错的呢？如果对错都分不清楚，做测试的意义又是什么呢？

（4）多个用例混在一个用例中。这也是刚入门的朋友容易出现的“好心办坏事”的情况，把测试用例写得特别长，包括了很多内容，这样很容易引起混淆，不如分开。而且，如果有多个用例混在一起，你的用例标题怎么写？另外，如果其中有几个用例通过，而另外几个没有通过，这时测试的结果很难记录，无论是把这个大的用例记录为通过或者不通过都不合适。

上面列出来的几个问题，大家可以尽量避免。实际上，写测试用例最难的地方是，如何把测试用例写得全面？这只能靠实践经验的积累了。你看完这节文章以后，可以拿记事本这个程序来练练，学着写几个测试用例，“看花容易绣花难”，所以要多试试。

4.4 如何执行测试用例

虽然在上一节中我们讨论了如何编写软件测试用例，但如果你真是一

位软件测试的入门者，你到单位报到后接手的第一项工作很可能是执行软件测试用例，而不是去编写。你不要因此而郁闷，这样的安排是合理的，因为你毕竟是个新手，执行软件测试用例是一个迅速熟悉当前测试工作的好机会，而且压力不大。因为在英语中执行测试用例是run case，所以有些公司把执行测试用例叫做“跑case”，想来也很形象。这也可以算是一种行话，你可以了解一下☺。

为方便讨论，我们以上节中的测试用例为例：

测试用例：验证记事本程序可以编辑中英文混合的内容。

测试步骤：

1. 运行记事本程序；
2. 切换到中文输入法，输入中文“学习编写”；
3. 切换到英文输入法，输入英文TestCase；
4. 保存文件，文件名为testcase.txt；
5. 关闭记事本程序；
6. 双击testcase.txt以打开文件。

预期结果：

1. 文件的内容是“学习编写TestCase”。

当我们面对这个用例的时候，我们首先要做的是清晰且正确地理解用例，不带半点含糊。测试的特点就是严谨，你来执行一个测试用例就是要贯彻用例编写者的测试思想，不能有误解或曲解，不能用自己的主观意志去代替原来的意思。例如，第一步“运行记事本程序”，你就应当清楚地知道“记事本”是哪个程序，如果有疑问马上问清楚，否则，如果真的把测试的产品都弄错了，一切就都白忙了，还浪费了时间。这个例子因为浅显，所以出现误解的可能性很小，而在实际的工作中，还是会有很多模棱两可的地方，这个时候我们不能偷懒，要勤学多问。

执行用例不能走样。例如，上例中的第二步，要求输入“学习编写”

四个字，如果你为了省事，拷贝了这几个字，每次都是粘贴过来，快是快了，却违背了“原著”的意思，这样是不可以的。用例编写者要求用输入法来输入，肯定是有道理的。如果你发现没有检测“粘贴”的测试用例，可以建议增加，但不能在执行的时候就偏离了用例的本意。说一个万一的事儿，如果这个软件通过了你的测试，发布给用户，用户却发现不能输入，只能粘贴，这个责任你能负得起吗？

大家可能都知道，做软件测试要细心，这个要求在执行用例的过程中表现得很明显。我们在执行一个测试用例的时候，不但要注意实际结果是否与预期结果是一致的，而且在整个过程中都要保持观察。例如上例中，如果第四步执行保存后，你发现文件名并不是自己输入的testcase.txt，这时你就应当停下来，因为这就是bug。

我们执行测试用例的目的是什么？就是发现bug，所以，我们在执行测试用例的过程中，要收集好发现的问题，不能有遗漏。在实际工作中，执行测试用例的过程一般都是紧张的，工作量很大，并不像我们今天在这里讨论的这么轻松，因为你要不停地往前赶，所以容易出现一些遗漏的问题。每当发现一个问题，我们都要做好记录，而不要总以为自己能记得住，好记性不如一个烂笔头。Bug是最能证明测试工程师工作成绩的东西，好不容易发现了，如果还被自己遗漏了，岂不令人懊悔？而且，还给产品留下了一个隐患。

前面我说过，执行测试用例是一个很好的学习机会。你可以在工作之余，去体会测试用例编写者的测试思想，而测试思想对于测试工程师来说是最重要的。你可以想一想，哪些测试用例是自己没有想到的？测试用例编写者的思维主线是什么？经过这样的琢磨，你对测试工作就会有进一步的认识和体会。你还可以尝试着去扩充测试用例，这是一个锻炼和提高自己测试能力的好方法。

如果你是在家里自学软件测试，我建议你找一个志同道合的朋友一起

学习，各自编写测试用例，然后去执行对方的用例，这样既增加了学习的趣味性，又有利于互相促进和提高。

4.5 如何提交一个bug

我们从软件测试的基本要求谈到如何设计测试用例，再到如何执行测试用例，现在终于谈到了收获的时刻——提交bug。因为提交bug最能体现软件测试工程师的工作成果，所以我们要认真做好这一步。虽然bug是测试工程师提交的，但是与bug打交道的人却很多，例如开发人员、项目管理人员、甚至客户，让他人清楚地知道问题的所在应当是bug报告的第一要求。那么，一个规范的bug报告应当包括哪些内容呢？

（1）清晰明确地重现步骤。任何人，只要按照这些步骤一步一步地做下去，就能重现这个bug，这对于开发人员定位问题至关重要。

（2）要有预期结果和实际结果的对比。什么是问题，而什么又是预期的正确结果，这是应当明确指出来的。

（3）级别定义。一般来说，把bug分为5级是比较合适的。至于1、2、3、4、5各代表什么意思，是可以自己定义的。更为规范的企业中，会把bug的级别区分为严重性和优先级两个不同的指标。

（4）如果有可能，可以做一个可能的原因分析。这个分析有的时候能缩短开发工程师修正bug的周期。

（5）如果是界面方面的bug，尽量附图，图片更有说服力。

（6）其他信息，例如软件版本号，测试环境的要求，测试工程师的名字，等等。

说了这么些要求，可能有点“干巴巴”的，下面我们来看一个例子。因为要在大家都熟悉的软件中找一个bug不是一件容易的事情，所以我这里做一个假设，假设有这么一个bug：在一款计算器软件中，做了乘法后把结

果清零，然后接着做加法，乘法的积累加到了加法的和中，清零的功能没有实现。下面我们规范地描述一下这个bug。

Bug：虽然已经对乘积做了清零，但是乘法后做的加法出错

软件：神算盘计算器

版本号：1.0.0.1

操作系统：XPsp2

级别：1（很严重）

测试工程师：测试新手

重现步骤：

1. 运行神算盘计算器软件；
2. 做一个任意的乘法，例如2乘以3，计算出积为6；
3. 点击“清零”。界面显示为0，这是正确的；
4. 做一个任意的加法，例如2加上3。

预期的结果：

1. 和应当为5。

发现的问题（实际结果）

1. 和为11。

原因分析：

1. 可能是因为点击“清零”后，虽然界面显示为0，但程序内部没有做相应的操作，导致乘法的积影响了后面的加法运算。

看起来这个bug很简单，但不代表在实际工作中都是这样一类简单的bug。在实际的工作中更大的挑战是，在沉重的测试工作中是否还能保持耐心，把每一个bug都描述得很清楚。

对于一位测试工程师来说，可能带来伤害的是什么？是假的bug。在任何团队里，如果一个测试工程师提交的假bug的个数稍高一点（要做到绝对不提交假bug也不大现实），会带来很大的负面评价。测试工程师在客观上

有一点监督开发工程师工作的味道，而如果这个“监督者”自身不过硬，又如何能让人信服呢？所以，发现了bug后，不要急着去提交，自己多做一次确认。不确定的bug，发现了真正的重现步骤后再做提交，也可以先去和同事讨论，以求发现真正的问题。在实际测试工作中，发现了bug后我都会先记录下来，等头脑冷静下来后再重现一遍，如果能重现，我才提交。我发现这个措施挺有效的。

另外，我们也要尽量避免提交重复的bug。一个bug，如果张三已经提交了，后来李四又做了提交，这既耽误时间，又会造成开发团队对测试团队的不良评价。我们可以尝试着养成一种阅读其他测试工程师的bug报告的习惯，这样头脑中就会大概有一点印象，对避免提交重复的bug很有帮助。

Bug提交了以后，并不意味着万事大吉，我们还要对bug保持跟踪，而这就是下一篇的内容了，欢迎你继续保持关注。

4.6 对bug保持跟踪

在上一篇中，我们一起讨论了如何提交一个bug，我在篇末说到，bug提交了以后并不就是万事大吉了。那么，后续还有什么事情呢？在回答这个问题之前，我提出另外一个问题：软件测试的目的是什么？是发现bug吗？

的确，发现bug是软件测试的目的之一，但不仅限于此，我们还要让这些bug得到妥善的处理，这就要求我们在提交bug之后还要跟踪它们。如果你在提交它们之后就不闻不问，会对整个测试工作造成延误。

在你提交了一个bug后，如果开发人员完全认可你的观点，修正了问题后又返回来让你确认，你确认以后关闭这个bug，我相信这个流程从任何人来看都没有问题的。可是，在实际工作中往往没有这么简单，以下情况都有可能发生：

（1）开发人员不认同这个bug，例如他们认为这不是一个真的问题，或者不认同你定义的级别，或者认为不需要解决，这些都需要你及时响应，

再次审查或者强调自己的观点。当出现不同的观点时，不要心急上火，我们要尝试着去说服他们，争吵不能解决任何问题。

（2）Bug描述不清，导致开发或者项目管理人员理解错误或者无法理解。这个时候，你不要不服气，你自己写的自己当然能理解，而他人不能理解的原因肯定是在描述上有些不足，例如是不是有些条件没有写进去，是不是用了一些模糊的词句，是不是英语表达有问题，等等。

（3）开发人员需要更多的信息。例如，开发人员希望看到发现错误时的软件界面截图，或者希望看到系统日志，等等。为了协助开发人员尽快解决问题，推进项目进度，测试工程师应当及时提供这些信息。

在软件产品的研发后期，一般都是以bug来驱动开发，也就是说，以bug的“发现——解决”来推动产品的研发。做为测试工程师，我们要对自己提交的bug保持跟踪，及时响应，不让产品的研发工作在自己的手里被延误。球到了你脚下，如果你没有反应，不开踢，就是你的责任。

如果没有适时的bug跟踪工具，我们每天查看一次自己的bug集合是应当的。在时间允许的情况下，我们还可以看看测试组其他同事提交的bug，这样能拓展思路，并了解测试的整体进展情况。

4.7 性能测试

功能测试在前面已经介绍过了，我们现在来看看什么是性能测试？我的这篇短文只能对性能测试做一个简介，如果你看过我的这篇文章以后对性能测试感兴趣的话，你可以去买本书来看看，段念的《软件性能测试过程与案例剖析》（清华大学出版社）我觉得就不错。

性能测试是什么呢？我们来看一个实例。现在很多城市都设有火车票代售点，虽然收点手续费，但是比跑到火车站去买票还是划算，也挺方便的。春节前，我到附近的一家代售点去买票，发现有几个问题：

（1）代售点的售票系统售票很慢，主要是查询速度慢，糟糕的是有的时候还会出现程序无响应的情况，只有强行关闭售票系统再次登录才可以用，其操作系统还是Win98，折腾几次后操作系统受不了了，罢工，只能重启计算机。

（2）代售点的售票系统登录很慢，连接服务器花个几分钟是常见的事情。

经历过买票难的读者都能体会到，以上的情形在春运期间是多么耽误事儿，很可能就在这个售票点还在等待查询结果的时候，车票已经销售完了。这个时候，你吼几声也没有用啊，计算机听不懂你的埋怨。只要不是大的节假日，不会有这种情况，而在节假日出现大量购票需求的情况下，系统反应就慢了，这就是软件系统的性能出了问题。

性能，如果我们不去抠什么科学的定义，我们可以暂时把它理解为“处理速度”。从上面的例子中，我们知道在做性能测试的时候，一定要考虑到系统最繁忙的时候。例如，对火车售票系统做性能测试就要考虑到春运的情况，那时候是所有的售票窗口、售票点都开足了马力卖票，而且售票员操作的速度飞快，服务器的压力很大。我们做性能测试的时候，假设的客户端一定要比实际可能的数还要大。

把性能理解为“处理速度”只是一种形象的说法，其实这里面包含了很多软件方面的指标，性能测试要监控所有这些指标，例如：

（1）响应速度。例如，一次查询，从提交查询到结果显示出来需要多长时间？

（2）吞吐量。例如，服务器在一秒钟或者一分钟内能处理多少个请求？这里的请求是指客户端对服务器提交的处理请求，包括登录验证、查询、交易等等。

（3）CPU占用率。

（4）占用内存数和内存占用率。

如何做性能测试呢？实际上，做性能测试的第一步是定义什么是预期的性能，也就是什么情况才算达标。没有这个，性能测试就没有了指南了。这个定义要越详细越好，例如，不要简单地说："火车售票系统要允许300个用户同时使用"，而是：

（1）300个用户同时登录的时候，每一个用户登录过程不超过5秒；

（2）当系统内有300个活跃用户的时候，系统对每一个查询请求的处理过程不超过3秒；

……

性能测试一般都需要使用测试工具，不然如果你做一个测试就需要公司给你找300个人去操作，可能就要把公司做倒闭了。现在你对测试工具也可以做一个了解，不过不必着急，测试工具不会很难用，一般1、2天就学会了，难学的工具不会受欢迎的。

正如我在开头所说的，这篇短文对性能测试只做一个简介，希望读者通过这篇文章能对性能测试了解一二。

4.8 界面测试

软件测试工程师有一种思维上的优势，那就是考虑得多、考虑得全面，因为这样才有利于bug的发现。对于想要迈进软件测试大门的朋友来说，我们也要主动尝试着让自己在考虑测试问题的时候，尽量想得多一些，全面一些。前面我们介绍了功能和性能测试，它们都很重要，但这并不意味着测试就做完了。我们还要多考虑一些其他的，例如做界面测试（UI测试）。

在实际工作中，测试工程师的面试是我的工作任务之一，因为公司常年需要大量的测试工程师，所以我做了大量的面试。我有一道笔试题，是考察候选人测试能力的，我在题中给出了一个软件的界面，在这幅图中我故意把界面设计得有点问题，把界面中的按钮排列成了台阶状，一级一级

升高，而不是日常中我们常见的均匀分布。我希望候选人除了做功能方面的测试以外，能够发现这个界面上的bug，但是很可惜，95%的解答中没有提到这一点。

所谓界面测试就是在功能和性能测试之外，对界面设计、不同分辨率、默认焦点等做测试。在我的《软件测试实战－测试Web MSN》这本书中有单独的一章讨论这个问题。做为一个初学者，我们可以从以下方面去考虑界面测试：

（1）界面元素要符合通用的标准。我记得微软公司有一套界面设计的规范，我们日常使用的微软的操作系统、办公软件的界面都是符合这个规范的。一般来说，如果你测试的软件是在Windows系统上运行，一般都要遵守这个规范，一来美观，二来也容易被客户所接受。没有遵守规范的，我们就可以认为它是一个界面的bug。

（2）界面尺寸的改变（缩放）。因为界面的变化会带来几个相关联项的变化，例如控件要不要成比率缩小或者放大，控件要不要重新排列，界面刷新等，这些容易出现问题，需要做测试。

（3）操作系统层次上的不同外观设置。在Windows中修改操作系统的外观设置是很简单的，之所以要考虑这种情况，是因为不同的用户会使用不同的操作系统外观设置，看看我们的软件是否能否适应各种情况。

（4）不同分辨率。用户因为硬件条件和个人喜好而使用不同的分辨率，这也是一个容易出现问题的地方。例如，软件界面在分辨率是1024×768的情况下显示正常，而在分辨率是800×600的时候界面底部的一些控件就有可能显示不出来，导致用户无法使用。这是测试工程师需要考虑到的。

（5）焦点问题。每个界面都有焦点设置，这里我们需要考虑几个方面：默认焦点是否合适；执行一个功能后焦点变化是否正确；焦点跳变的顺序（用Tab键来验证）是否正确等等。

（6）对静态文本、图片和动画的检查。我们要去检查界面上所有的文

字、图片、动画是否正确，不能有错别字，不能有语法错误，更不能有违反法律的字句和图像存在。

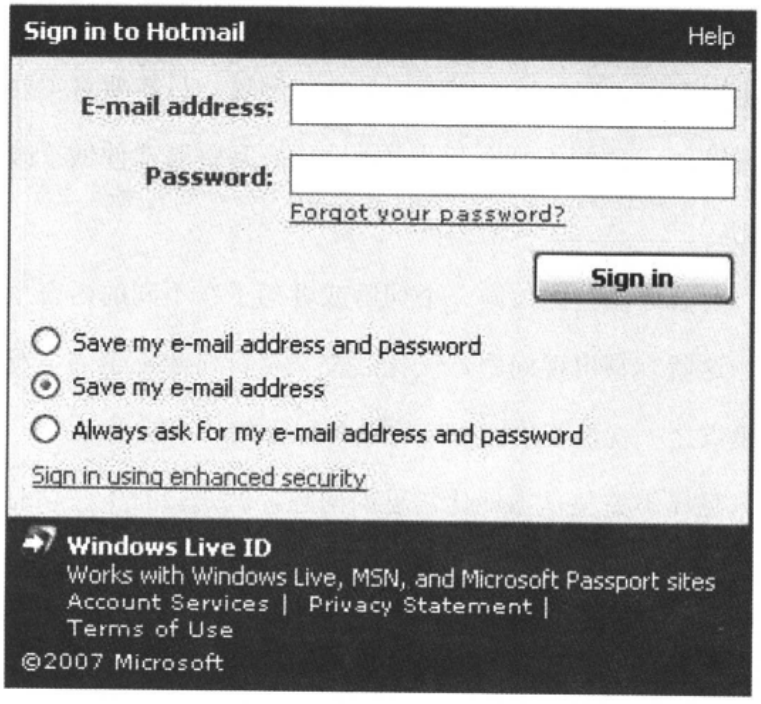
（7）快捷键。验证各个控件的快捷键是否正确。可能有的朋友对界面为什么要有快捷键有点疑惑，理由是：用户通过快捷键可以实现更快的操作；不是任何时候用户都有鼠标，也不是任何人都喜欢用鼠标。

其他的还包括界面风格、界面颜色搭配等等，我们都可以进行验证。界面测试不复杂，也不难，只是我们不要忘了做。

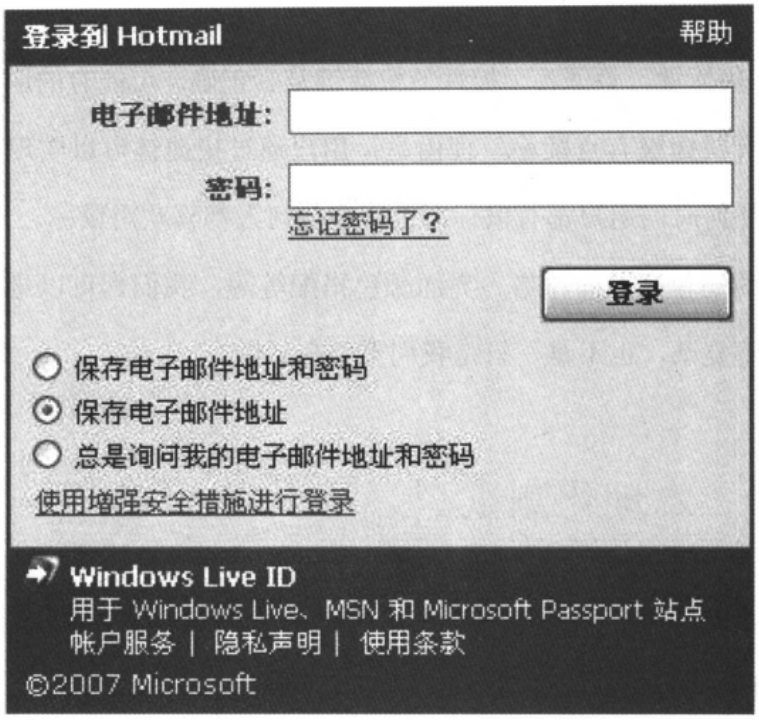
4.9 本地化测试

对于最初接触软件测试的朋友来说，本地化测试与我们日常所说的测试有些不同。在讨论什么是本地化测试之前，让我们来先看看什么是本地化。本地化实际上就是翻译，就是国外的软件经过汉化，拿到中国市场来销售，例如Windows中文版、Office中文版等。在汉化的过程中，测试工程师需要介入，做一些验证。

我们来看一个例子。下图是Hotmail的英文登录界面的一部分：



下图是Hotmail的中文登录界面的一部分，是汉化的结果：



大家可以看到，这两个界面整体上是一致的，也有些许不同，例如字符串的长短、某些控件的大小、字体、排列位置等。那么做本地化测试的重点是什么呢？

第一，要验证主要的功能是否正确。例如Hotmail的登录界面，汉化后最基本的也是最重要的是要实现登录功能。如果连这个功能都出错了，界面再好看也没有用。但本地化测试不是功能测试，只需要对主要的功能做一个验证就可以，因为从理论上讲，本地化只是资源文件做了改变，源代码没有变化。

第二，要看界面。因为同一个词语或者句子在不同的语言里的长度是不一样的，这就容易出现问题，汉化后中文字符显示不全的情况很常见。本地化的难点之一就是要验证所有的界面和窗口，不能遗漏。

第三，翻译不完整是本地化常见的问题，就是会有一些信息没有被翻译。这个问题和开发的规范性很有关系。如果开发的流程比较规范，所有的字符串都存放在一个资源文件里，出错概率小一些。如果要开发人员到

源代码里去寻找需要翻译的字符串，因为要翻译的字符串很分散，这种情况下翻译不完整的可能性很高。

第四，翻译出错。虽然从理论上说测试工程师不应当对翻译的正确性负责，毕竟术业各有专攻，但是如果顺带着也能发现几个翻译的错误，当然更好了，还可以练练英语☺。

现在因为中国市场巨大，对国外很多的软件厂家都很具有吸引力，致使软件本地化业务在增长，所以也就提供了越来越多的本地化测试的职位，大家对本地化测试做一个简单的了解，求职的时候也可以做为一个考虑。

4.10 测试的根本：正确理解需求定义

对于一位想自我提高的软件测试工程师来说，可能有的时候会有点疑惑，疑惑什么呢？标准太多了。现在资讯发达，各种文章触手可及，有的文章说“软件测试工程师应当……”，有的文章说“优秀的软件测试工程师不应当……”，说法多样，五花八门。在这里我根据自己的测试体会，提一点自己的看法：做好测试的根本是要正确地理解需求定义。

测试的任务是对照需求和实现发现软件其中的错误。这里的“需求”有多种，并不只局限于需求规格说明书。

测试工程师需要理解的需求有：

（1）产品的需求规格说明书。这肯定没的说，如果这个都不了解，测试没法做了。

（2）产品需要遵守的软件规范。虽然没有明文规定，但是软件都有一些规范，例如界面的设计、控件样式、排版格式、命令格式、文件名等等，这些在业界内都有一些通用的做法。这些规范可能并不存在于客户的要求或者需求文档里，但如果没有遵守这些做法，就会显得很不专业。

（3）易用性上的要求。用户怎么用起来更方便，这是易用性上的要求。

以前，软件产品是“大腕”，要求用户要小心翼翼，用户输入错误或者操作错误导致软件出错，责任在用户。这样的“好时候”已经过去了，现在用户是“老爷”，用户要觉得不好用，即使你的功能并没有错，用户还是会抱怨，或者要求修改。

现在很多房子都有可视对讲器，一个单元装一个，每家都有终端。你的朋友来了，在单元门前输入你家的门牌号就可以呼叫你，你能看到来访者的视频图像并实现对话，可远程打开单元门，即方便又安全。我住的房子的可视对讲器还挺有意思，它一共可以输入四位数字，问题是它要求每次都必须输入四位。我们这栋楼只有6层，门牌号都是三位，它要求不足四位的前面补0，例如门牌号是302，只有输入0302才能实现正确的呼叫。就是这个独特的要求，有几次我的朋友都是在单元门前给我打电话，说这个对讲器怎么不工作了。我觉得这是个软件设计上的缺陷，测试工程师也要承担责任，从一个普通用户的角度来考虑，他怎么会知道门牌号码不足四位的前面要补0？是要考智力题吗？这种要求就是易用性上的基本要求。

在我们测试工程师充分理解了产品的需求定义之后，包括纸面上的或者默认的规范，我们就可以开始编写测试计划和测试用例了。测试的执行有难有易，但那是技术层面上的事情了，首先我们要想到需要测试什么，彻底地理解需求会让我们解决这个问题。

我们可以看到，软件需求规格说明书对测试工程师有重要意义，可以说是软件测试的“红宝书”，所以，没有软件需求规格说明书的测试项目会比较难做。如果软件的需求只是在客户或者开发人员的脑袋里，这是对软件开发以及测试很不利的一种局面。如果真的遇到这种情况，一方面我们要据理力争，向管理层反应这个问题的严重性；另一方面我们可以自己尝试着做一些总结，把会议、邮件甚至对话当中大家达成的共识总结出来，总能描绘出软件需求的一个大体框架。

多看几遍软件需求规格说明书，这是基本功，好好练☺。

4.11 测试参与到研发的各个过程

一旦参与到了实际的软件测试工作中，我们就要树立一个信念：软件测试要参与到研发的各个过程。以前在软件工程的理论当中有一种比较生硬的划分方式，把软件测试算作编码后的一个阶段，其他阶段则没有测试什么事，这是不对的。

在软件需求规格说明书的编写阶段，测试人员就要参与进来，参加各种需求评审会议，审核需求文档。大家都知道，需求规格说明书是软件开发的“总体规划”，是研发过程中的“纲领性文件”，而它又是人写的，不是神仙写的，自然会有错误。如果我们测试人员能够发现需求规格说明书中的bug，那就是立了大功。例如，需求中有一个功能A，经过设计、编码，最后通过测试才发现这个功能是完全没有必要的，如果能在需求评审的时候就发现这个问题，去掉功能A，省了不少力气，又降低了研发成本。另外，紧密地参与到需求编写阶段，能让测试人员对软件产品有一个深入的了解，这样才能写出好的测试计划和测试用例来。

软件设计相对来说是开发人员的“内部事务”，有些开发人员也不愿意测试人员参与进去，但是测试人员的参与是很有好处的。如果测试人员对产品的需求很熟悉，而不了解产品的设计，这个产品在测试人员面前始终是一个黑盒子，不知道里面是怎么回事。测试人员了解了产品的设计以后，黑盒子就变成了灰盒子或者白盒子，测试人员就能够有目的性地设计出一些测试用例，来发现比较深入的问题。而如果测试人员能在软件设计阶段发现问题，那是很受欢迎的事情，这样能避免软件研发走弯路。

软件发布以后，就进入了维护阶段。软件公司会根据客户的反馈，修正一些bug，或者对一些功能做改进。不要在修改以后就想当然地去发布软件补丁，软件测试人员对修改的部分的确认是很有必要的。虽然发布补丁总是基于让软件产品表现得更好的想法，但是不要好心办坏事。软件测试

人员的参与，能有效避免软件补丁“开历史倒车”的事情发生。

积极地参与到软件研发周期的各个过程中，是软件测试的职责所在，是一位软件测试工程师应当做的事情。

4.12 测试需要一个支撑平台

测试工作有一个特点：重复。例如一个测试用例写出来了，在不同的时间会被重复地执行。同时，我们需要记录每次的执行结果。如果一个测试用例被执行了10次，这10次是通过了还是失败了，是因为哪一个bug失败的，这些都要记录下来。在记录大量信息方面，计算机比人要优秀得多，所以，在做测试之前，我们需要建立一个测试支撑平台。

首先，应当有一个测试用例管理工具，用来存储测试用例以及执行信息。如果项目小而公司又财力有限，可以只把测试用例存放在文档（例如word）中。如果项目稍大一些，就应当考虑部署一个测试用例管理工具，因为文档中的信息很不方便统计和查询。如果测试组不能及时提供测试统计数据，对项目是不利的。因资金紧张而不能购买的话，自行编写一个也是可以的。一个测试用例管理工具需要记录哪些信息呢？

（1）项目名称；

（2）模块名称；

（3）测试用例名称；

（4）测试用例的具体步骤；

（5）测试用例的优先级；

（6）编写人；

（7）每次的具体执行结果，例如，通过，还是失败。

其次，应当有一个bug管理工具。测试人员发现了一个bug后，需要把它公布出来，等开发人员修复后再做验证，验证修复正确后就关闭这个bug。

项目组各方面的成员都会参与到这个过程中来，没有一个大家都能使用的工具势必导致效率很低。自行开发一个bug管理工具是完全可能的，它至少要记录这些信息：

（1）Bug名称；

（2）严重等级；

（3）优先级；

（4）所属项目和模块；

（5）发现的步骤和具体表现；

（6）发现时间；

（7）产品版本号；

（8）Bug的状态（例如，是打开，是修复，还是关闭）；

（9）备注。

再次，应当有一个测试报告工具，用来统计测试数据。之所以把测试报告算作测试支撑平台的一部分，是因为我希望这个报告能做成一个站点或者一个工具，大家随时可以访问，了解进度和结果。如果没有这种条件，只能辛苦测试组长了，由他（她）定时给大家发邮件吧。

最后，不要忘了，独立的测试环境也是很重要的。不要被“独立的测试环境”吓住了，环境有大有小，财力不足的，几台计算机也是一个测试环境，但是，重要的是“独立”。让测试环境只做测试，没有任何其他任务。不能拿工作计算机来做测试的主力，测试环境的混乱必然会影响测试结果。

有了良好的软件测试支撑平台，我们就能放开手去做好测试了。

4.13 如何考虑得更全面

我首先必须承认，写这篇“如何考虑得更全面”对我是一个挑战。

我一直认为，考虑全面是测试工程师的价值体现。在一个项目中，技术难题由开发人员来扛，软件构架、具体开发细节上的难题不会落到测试人员的肩膀上。在一个技术性的团队里，没有一点专长是不行的，从角色分工上来说，团队需要测试人员考虑全面，在产品发布前尽可能多地发现bug，使产品以满足市场的要求。

我们以一个拨号程序为例来说明。

对于开发人员来说，他要考虑的是：

（1）调用什么系统函数能实现拨号；

（2）写一个什么逻辑能保证拨号成功。

对于测试人员来说，他考虑的就要多得多：

（1）功能上：是否正确实现拨号功能？掉线后能否实现自动重拨？能支持多少种拨号设备？

（2）性能上：拨号速度如何？占用内存和CPU是多少？长期运行是否稳定？是否有内存泄露？

（3）界面上：界面设计是否符合规范？字体、颜色的设置与搭配是否恰当？

（4）易用性上：是否符合用户的操作习惯？是否支持快捷键？在各种情况下，是否有简明正确的提示？

（5）兼容性上：是否兼容各种常见的操作系统？是否兼容各种常见的软件？

（6）安全性上：保存的用户名和密码是否容易被盗取？

……

如何让我们软件测试人员考虑得更全面呢？我根据自己的经验，提出以下几点供大家 参考：

（1）自己多冥思苦想。不管其他条件如何，我们的测试工作做到什么程度，基本取决于我们自己。在设计每一个功能点的测试用例的时候，鼓

励自己多想，而不是浅尝辄止。思想就好像一根钉子，我们自己多加把劲，它就能钉得深一点。如果自己不努力思考，光想着灵感光顾或者外援支持的话，是不可能做到全面的。

（2）多评审。通过实践，我觉得把自己的测试计划和测试用例拿出去做评审是一个进步的好机会。每个人看问题的角度各有不同，通过开评审会，特别是进行热烈讨论的评审会，我们都能从同事的发言中看到自己的不足。看到了不足后的补充和改进不是一件难事，关键是去如何看到不足，评审会能给我们这个机会。所以，不要掩藏和退缩，只要有评审的机会，就要积极响应，这是让大家放下手中的工作为你“打工”的机会，不要错过。

（3）多看同事的测试文档。在一个团队中，会有并肩作战的同事，他们的测试计划和测试用例是学习的好材料。我们可以通过阅读来揣摩其中的思路，汲取其精华，补充和拓展自己的测试思路。

（4）多看书。作者能写一本书出来，不管如何，肯定有一些可取之处。现在测试方面的书籍不少，虽然书中所讲的很难与自己手头的项目一致，但是思路和方法是可以借鉴的。

（5）多参加讲座。只要是技术讲座，无论是开发还是测试方面，只要有时间，都可以去听听。有的东西看起来离自己的工作有点距离，但很有可能你听着听着就会想到自己的测试工作，想到“如果我怎么怎么做就好了。”，这就是一个收获。听讲座还有一个独特之处，就是可以与主讲人现场沟通，这有利于我们迅速为自己的疑惑找到答案。

以上种种，都是我从工作实践中总结出来的，我相信还会有其他有效的方法通过工作实践被总结出来。

4.14 版本控制

如果没有版本控制，对测试工作将是一场灾难。

版本（build），就是用户将会看到的产品，例如我们通常看到的可执行文件，动态链接库，等等。如果开发人员随意地做出一个版本，扔给测试人员，说，测试吧，就好像说："嗟，来食！"。测试是需要成本的，这种嗟来之食我们不能吃。不稳定的和没有控制的版本就是一片沼泽，很容易陷进去而无法自拔。

如何做好版本控制呢？我给出一些建议以供参考：

（1）我们只需要开发人员提供源代码，而不需要他们去做build。也就是说，由测试人员或者第三方（例如专门做build的人）来做一个新版本。这样，程序版本就脱离了开发环境。

（2）我们要对每一个新版本做BVT（Build Verification Testing，俗称冒烟测试）。没有通过BVT测试的版本，测试组一律不予接收。对于在BVT测试中发现的问题，开发小组应当立即解决。

（3）选择一个稳定的版本来做测试。如果一个项目有1000个测试用例，我们当然不希望做到第300个用例的时候才发现版本有重大缺陷，导致测试无法进行。在大规模开展测试之前，我们需要做一个版本选择的测试，这种选择由测试用例的一个子集来确定。例如，我们从1000个测试用例中选50个最重要的用例，只有通过了这50个用例的版本才会被采纳作为测试版本。这种测试可以称为CC（Code Complete）测试。

（4）为了跟踪和后期分析方便，任何一次的测试结果中都要有版本信息，例如，1.0.0.1234。有了版本信息，我们可以知道哪一个版本稳定，哪一个版本不行。这个版本信息对于开发人员定位和解决问题也是很有用的。

（5）在开发人员向代码服务器上传最新的代码之前，如果测试人员有精力的话，可以帮忙做一下确认测试。例如，开发人员的这次修改是为了

修复一个bug，他在自己的开发环境下试了，结果OK，测试人员再帮忙在测试环境中试一下，减少开发人员提交错误代码的概率。

现在一个项目的开发周期都不是三天两天，版本控制要做到有始有终。如果你说这个项目我做了前半段的版本控制，后面就放羊了，这一点意义都没有。做好版本控制，对开发和测试都有重要的意义。

4.15 如何编写测试计划

我记得自己在刚开始做软件测试的时候，工作了大约一两个月以后，要写一个测试计划，这是我写的第一个测试计划，心里有些紧张。我上网找资料，到书店找书，特别是想找模版和实例，我感觉写测试计划是件重要的事情，希望能找到一些相关的资料和比较好的模版，这是当时的真实感觉。

“不当家不知柴米油盐贵”，写测试计划就是要当一个测试项目的家。如果你是写一个模块的测试计划，就是当一个小家；如果你要写整个软件产品的测试计划，就是当一个大家。这家可不好当，有很多方面需要考虑：

（1）测试的范围。要测试什么，这是肯定要明确的。即使你知道，你也要写出来，让看这份文档的人知道测试的范围。在确定测试内容的时候，还可以做一个优先级的区分，这样能保证工作是按照优先级的高低来推进。另外，什么是测试范围之外的，也就是不需要测什么，也是需要明确的。

（2）测试的策略。测试的策略是指我们测试的指导思想是什么。例如，手工测试和自动化测试的分工，黑盒测试和白盒测试的分工，等等。

（3）测试的思路。也就是针对具体的功能点或者性能指标，我们做什么考虑。这一块是重点，我认为可以写详细一些，可以把测试用例（此处不写用例的详细步骤）都包含进来。

（4）测试进度的安排。

（5）测试资源的安排。例如，人员、计算机设备、软件等等。

（6）风险分析。在我们刚开始做软件测试的时候，一般都是去执行别人编写好的测试用例或者写一些测试用例，那个时候我们一般都不去考虑什么风险。而在写测试计划的时候，我们看问题的层面就要提高一层，既要考虑怎么去做测试，也要去考虑什么是测试的风险，也就是容易遗漏或者失败的地方，或者测试安排上的一些为难之处，例如时间紧张等。认识到了风险，我们就能采取一些措施有效地防范风险。

没有解决的问题。软件研发是一个迭代的过程，任何时候都有悬而未决的问题，或者仅仅是你有疑问的地方。我们要把它们罗列出来，这有利于问题的讨论和解决。不然的话，如果漏掉了，就会成为一个隐形的炸弹，不定什么时候爆炸，对工作产生干扰。

如果以上的问题都有了答案，我们就找一个漂亮的测试计划模版，把这些内容填进去，一个测试计划就成型了。测试计划的格式不是重要的，更重要的是内容。

4.16 测试任务分解后的风险及其应对

现在测试组只有一个人的情况越来越少了，因为软件产品的要求越来越复杂，一个人单挑测试活儿的可能性很小了。规模大了，根据经济规律，大家要分工协作，每个人都有自己的一亩三分地，每个人的工作范围小了，效率也就高了。但是，这种工作方法暗藏着一个风险。

例如在一个测试小组里，小李负责模块1，小赵负责模块2，工作都很努力。可是，谁来负责两个模块关联的部分呢？可能会有一个bug，是在用户使用了模块1之后再使用模块2时出现的，这个问题可能小李、小赵都忽视了。我们不可能说再派一个人来负责模块之间的关联部分，如果你提出这个想法，就准备面对老板的冷面孔吧☺。

这种“三不管”地带会是产品质量的隐患，作为测试工程师，消除这种隐患也是我们的工作职责。软件质量适用“圆桶定理”，最短的那块木板会是软件产品的真正质量水平。只要出现软件错误，就和所有的人相关。只有软件质量高，测试组的工作才会有成绩，个人才会有成绩，产品不会只因为一个模块好而得到好评。

如何来降低这种因任务分解而带来的风险呢？我们可以尝试以下的方法：

（1）除了自己负责的模块之外，还要了解整个产品的细节。如果你负责模块1，同时也需要了解和熟悉模块2，模块1和模块2不会是老死不相往来的。

（2）去阅读其他人的测试计划和测试用例，拓展自己的测试思路。在时间允许的条件下，可以做一些交换测试的尝试。例如，在研发期间的一轮测试中，你可以建议leader安排你去执行小赵编写的关于模块2的测试用例，小赵来执行你编写的测试用例。这样交叉运行测试用例是熟悉其他模块的好办法。

（3）多考虑用户的实际使用场景。用户不可能长年累月地只使用某一个模块，用户做一件事情，可能要用到多个模块。测试工程师对这些使用场景的假设能力是很重要的。考虑这些使用情况，模块之间的问题就会被测试人员所发现，“三不管”就会少得多。

（4）安排一些探险测试。所谓探险测试，就是大家抛开测试，完全按照自己的意愿去使用软件产品以期发现错误。这种做法主要是为了打破大家的思想局限，发现一些以前没有注意到的问题。

你在实际的工作中可能还会发现一些有效的方法来降低任务分解后带来的风险，如果愿意，欢迎与我一起分享。

4.17 实施自动化测试

现在越来越多的人谈论自动化测试，为它的“自动”所吸引。而它对于中小软件企业来讲，的确是一个不小的挑战。我们已经意识到了应当去做自动化测试，因为它能够提高测试效率，降低测试成本，是软件测试的一个趋势。但是，限于人力资源和测试工具资源，自动化测试如何才能开展呢？像IBM或者微软这样财力雄厚的企业毕竟是少之又少啊。

在做自动化测试之前，我要泼一盆冷水。在实施自动化测试的前期，测试的成本非但不会减少，反而会增加，因为你需要采购或者编写测试工具，需要去尝试，需要额外的人力资源。只有当自动化测试进行到一定阶段，降低测试成本的效果才会显现出来。我之所以泼这盆冷水，是怕你对自动化测试有过高的期望。

自动化测试包含很多种，并不只局限于Robot那样的录制回放工具。从成本的角度考虑，我的建议是先将性能测试或者压力测试实现自动化。性能测试自动化是手工所不能及的，例如，你公司的产品是人力资源管理系统，需要检查当有200个客户端同时使用这套系统时系统的表现，你不可能雇200个人来做这项测试吧。而这项测试通过一个工具很容易实现，然后一台计算机、一个人就够了。我们让测试工程师编写一个工具，模拟200个客户端去连接服务器，做指定的操作，然后取得服务器端的性能数据。如果测试工程师面露难色，没有关系，指派一个开发人员去给他讲解一下产品的某些实现细节，既然产品的客户端可以连接上服务器端，测试工具为什么不可以呢？这种自动化测试的成本低，不需要采购，也马上就能见到效果。

实现了性能测试自动化以后，我们接下来可以考虑做BVT。当产品有新的build出来，我们要尽快做BVT以知道这个build是否可用。如果每天凌晨两点出build，随后BVT能够自动进行的话，早上一上班我们就能看到测

试结果，这样的话测试效率得到了提高。BVT一般都是功能测试用例，大多UI打交道，没有工具的话自动化测试没有办法实现。BVT的测试用例一般都很少，容易做完。我们可以通过使用某一个测试工具的评估版（免费下载，可使用期限各不相同）来尝试做BVT，可以在评估时间内得出一个是否购买的结论。

最后要做的当然是全面铺开，让更多的测试用例自动化。因为有了前面的铺垫，相信到了这一步会势如破竹，实现自动化测试只是一个时间问题。

自动化测试的维护也存在一个工作量比较大的问题。产品的变化会导致自动化测试代码也要发生变化。例如，如果客户端和服务器端的协议变了，性能测试工具就一定要改变。如果界面做了调整，UI自动化测试就要改进。在这里，管理层要对开发人员提出一个要求，在做了一项变动后要尽快通知测试方，以避免测试工具失效。

实际上，自动化测试的实施是一个迭代的过程，一步一步地做，一步一步地扩大，根据自己公司的实际测试要求和经济状况，寻找一条适合自己的路。

4.18 改进测试的几种方法

即使你已经意识到了测试的重要性，但很有可能你对测试工作的现状不满。虽然测试的门槛低，但是做好测试是很难的，似乎测试工作总是不尽人意，甚至在有的公司背上了一个“花了钱不见效果”的恶名。如果你有了如何才能提高测试的疑问，我根据微软的常规做法给你几点建议吧。

1. 整个软件项目小组审查测试计划

在微软，测试计划编写完后，需要提交整个项目小组来审议，审阅测试计划是PM（Program Manager，程序经理）、Dev（Develop，开发工程师）

和Tester（软件测试人员）的工作内容之一。参加评审会的每个人都会对测试计划提出自己的看法和改进意见，一场评审会下来，测试计划的编写者都有大汗淋漓之感，同时也很有收获。通过评审会，测试计划就融合了大家的智慧，测试计划的质量也提高的一个层次。一个人再能干，考虑问题都会有不周到的地方，甚至会有大的遗漏，测试计划评审会是提高测试计划质量的有效途径。

那么，怎样让评审会开得有效果？测试计划编写者要提前几天把会议材料发出来，让大家有准备的时间。被评审的计划一定要在会前看完，如果开会的时候才来匆匆浏览一遍，怎么能发现问题？评审会只是为了给出时间对有疑问的地方进行讨论。在开会前，编写者应当收集所有的反馈意见，据此整理出讨论列表。例如在微软内部，如果你没能对一份计划提出改进建议或者建议很少，你会觉得很惭愧。这种惭愧代表了一种很好的工作氛围，这种氛围需要各级管理者去营造。

2. 请开发人员评审一部分测试用例

让开发人员参与进来，从他们的角度来看测试用例。有不同意见的时候，就是测试用例升华之时，这里强调的也是测试的公开性。因为开发人员的主要工作是开发，要求他们评审所有的测试用例是不可能的，只能挑选出重要的测试用例来做评审。同时，相信参加评审测试计划和用例对开发人员也是一个提高的过程，他们就会发现以前他们没有在意或者没有想到的地方，拓宽了他们的视野，无形中减少了bug的产生。

3. 做 Code Complete（编码完成）测试

如果开发人员扔给测试人员一个很差的或者缺乏一些主要功能的build，对测试工作是一场灾难。在微软，为了预防这种情况，特别定义了一个Code Complete测试。开发工作大体完成的时候，开发小组主动申明，然后测试小组要做一个测试，来验证开发是否真的达到了基本要求，就是验证代码编写是否真的完成。在做Code Complete测试的时候，是很有意思

的。因为这时候“球”还在开发人员那边，如果没有通过Code Complete 测试，对开发小组来说压力是很大的，所以一有妨碍Code CompleteC测试通过的问题，他们会马上修正，以求符合要求。这个时候要把握一个度，Code Complete只测试主要功能，点到为止，不要要求太高，因为后来还要进行详细的测试。一般来说，执行优先级为0和1的测试用例即可。

4．做好 BVT

开发人员在新增功能、代码优化或者修改bug的时候，可能会带入其他的问题。为了避免功能出现重大缺陷的build对测试造成的危害，就需要做BVT。没有通过BVT的build，表明它连最基本的要求都没有达到，这个build不能被测试组采用。

BVT中发现的任何bug都将导致BVT Fail（BVT失败），BVT Fail在微软是一个很严重的问题，开发人员需要在24小时之内解决。

5．正确定义测试用例的优先级

测试用例的集合总是很大，而且越详尽越好，但是，执行测试用例的时间是有限的，这就需要我们定义用例的优先级，做到主次分明。例如我们可以定义：

P0：最基本的重要功能。例如，在计算器程序中，我们可以定义验证最简单的加减乘除4种运算的测试用例为P0；

P1：重要的功能或者性能；

P2：细小的功能或者性能要求；

P3：少见的场景。

这样，我们就可以定义什么时候该执行什么优先级的用例，而不是每次都把所有的用例运行一遍，那样成本太高，而且测试人员会疲惫不堪。

6．要记得做性能测试

性能测试很难，但不可遗漏。通过性能测试，你可以知道服务器端可以支持的连接数，资源消耗情况，什么是瓶颈，等等，然后从中得到改进

的思路。你的产品功能再好，却不能容纳多个用户同时使用，或者执行速度很慢，这些都会导致软件项目的失败。而且，性能是很难提升的，需要给开发人员时间去调查和思考，所以越早发现问题越好。

7．做一些探险测试（Ad-hoc）

测试计划和用例编写得再详细，能囊括一切吗？当然不能。能奢望项目一成不变吗？当然也不能。所以，请在测试计划和用例之外，再留一点做探险测试的时间。抛开测试计划和用例，按照自己的理解去测试软件产品，相当于随意地使用产品，我敢肯定，你能发现自己以前没有发现的bug。

8．保持改进

不要认为自己的测试计划和用例已经尽善尽美了，要保持清醒的认识，任何时候，我们都是有可以改进的余地的。

4.19 更专业一些

在任何行业，如果你得到“做得很专业”的评价，这会是一种很高的褒奖。怎么才能把测试工作做得更专业？这个问题很难回答。我在这里讨论这个问题，并不代表我就是专业的。我仅提出一些观点，供大家参考。

（1）少提交虚假的bug。

（2）少提交重复的bug。

（3）穷追不舍。有的时候，一个bug可能需要测试人员去反复证实，去和PM、Dev交流，去推动问题的解决，在这样的过程中如果没有一股穷追不舍的劲儿怕是会草草收场的。更有甚者，有些功能可能PM考虑得不大全面，需要测试来驱动开发。

（4）重视交流。与PM和Dev多做交流，这样测试人员才不会孤陋寡闻。同时要记住说话的态度和方式，不要咄咄逼人。我的一个建议是，要让别人听得进你的“逆耳忠言”，你至少不要作兴师问罪状。前面有一个引子

比较好，赞扬尤嘉，然后慢慢道来。这里说的赞扬不是吹捧，而是从他人做得好的地方说起，然后再谈及bug，这样会比较容易被别人接受。

（5）分析错误原因。我们发现了问题后，如果能够得到日志，不妨我们先做一个日志检查，并把初步的问题原因分析写在bug报告里，这样不但节省了开发人员的时间，也会增长测试人员的经验。

（6）发现了问题后要再现一遍，不要急于提交。

（7）不要放弃不能重现的bug。有的bug具有偶然性，你遇上了一次，而重现的时候却再也看不到它。测试人员可能会不敢提交这个bug，因为没有办法重现。但即使只出现了一次，至少说明可能有问题，所以，不要轻易放弃，要记录下来，下次再次遇到，证据就全了。不可重现的bug也可以提交，只是要注明这个bug不容易重现，以免他人误会，这样既不浪费开发人员的时间，也不让一个bug溜掉。

（8）Bug的描述要清晰，无歧义。

（9）图片要经过裁剪。不要什么bug附带的图片都是全屏的截图，这样的文件打开速度慢，而且没有主次，开发人员不知道看哪里。图片经过裁剪后，重点突出，一目了然，节省了浏览的时间，何乐而不为呢？

（10）大局为重。测试人员总是希望开发人员能够改正所有的bug，让软件产品完美地呈现在客户面前，但这种想法往往不现实。在商业竞争中，产品的上市是一项重要的商务策略，测试人员应当服从集体的决定，不要纠缠于某一个具体的问题。

“做得专业”的评价是一枚闪光的军功章，需要我们付出辛勤的汗水才能得到。

4.20 英语——测试工程师应当掌握的外语

英语的重要性是不用多说了，至少到现在为止，计算机还是一个英语

的世界。对于英语每个人都有自己独特的学习方法，不会存在适合于每个人的通用方法，我这里介绍一点自己的体会，供你参考。

在得知拿到了微软公司的vendor职位之后，我老老实实地到联邦软件去买了《金山词霸》，因为我知道，在工作语言是英语的微软，它就是我的拐杖。虽然花在学习英语上的时间不少，从初中到大学，毕业后工作的业余时间也会看看，可是远远没有到达“熟练掌握”的程度，虽然简历里大家都这样标榜自己☺。

在《金山词霸》的帮助下，对于英语的读写勉强还行，因为读写方面我有很大的回旋余地，不会的我可以去查词霸，能够大概弄懂意思或者凑出一个句子。最大的挑战在听和说方面，听，需要实力；说，需要勇气。在项目初期，开会的时候很紧张，因为我们是和美国的同事开电话会议，我总是盯着电话，好像这样就能帮助提高听力似的☺，可最后还是没有明白。特别羡慕从美国回来的海归，说英语特流利，有的时候我还没明白第一句，人家已经把一段话讲完了。

我自己认为，对英语学习帮助最大的还是使用。例如，所有的邮件、文档都是英文的，你要写的东西也必须是英文的，这种情况对英语是有很大的促进的。以前也背背单词，可是总不用，几天后就忘了，总是这样在原地踏步走，没有效果。所以，我想，你可以先在工作或者生活中去使用英语，这对英语的学习很有意义。

在学习英语感到困惑的时候，有时我会想，母语汉语是怎么学会的呢？从依呀学语开始，先学会了说和听，然后才是进学校去学习读和写。所以，我就努力去找有助于学习英语听说能力的环境。项目组里有一位外国友人，他是另一位同事的导师，他们每周二有一次会餐会，就是大家在一起吃个饭，随便聊一聊。我觉得这是一个很好的机会，就申请加入。很感谢，他们接纳了我，这样我每周都有了一次直接用英语交流的机会。而且因为是吃饭的时候，非正式场合，所以压力不大，听不懂或者说错了都没有关系。

有一段时间ATC（微软亚洲工程院）从美国请了一位临时的英语老师（三个月），我很高兴，和他约了定期的见面时间，我们给对方介绍一些自己国家的情况和风俗传统。我觉得这些和外国朋友的接触活动对英语学习很有帮助，就好像在重新渡过自己小时候学说话的日子。时间长了，对英语就有了一点语言感觉和习惯，对一些简单的对话应付自如了，组织英语句子的速度也快了。身边没有这个环境的朋友可以通过英语角来完成这个过程，就北京而言，我觉得人民大学的英语角就不错，人多，气氛好。

另外，看英语影片对英语的学习也很有帮助。从简单有趣的看起，慢慢积累。之所以推荐看简单有趣的，是因为英语学习是一个长期的过程，如果看的影片过于沉闷或者严肃，没有乐趣可言，怕是不容易坚持下去。我想，学习的过程不是要去找最难的学，而是找最容易接受的。动画片或者一些肥皂剧，都是不错的选择。我一般是一边看一手拿着一个电子词典，碰到对句子中关键词语不理解的地方，就暂停下来查一下，弄明白后随手记录下来，然后继续。看英语影片主要是培养英语语感，我的英语主要是应用在计算机上，看影片对生活英语是一个很好的补充。我不知道你是否有这种体会，或许你可以用英语就项目情况写一封很长的邮件，但却难以描述日常生活中的一件小事，例如周末自己做了一个什么菜，偶然遇到一个朋友聊了聊。

按照上面说的方法我学习英语，过了一年多，我觉得还是不扎实。也尝试着买过英语报纸，但是没有坚持下来，不知道是因为报纸上的新闻没有激发自己的兴趣，还是自己没有毅力。现在我发现有一本英语杂志《英语世界》挺好的，小巧的32开本，里面都是一些短小精悍的文章，中英文对照，对重点难点还有批注，我觉得挺适合我的。

微软亚洲工程院为了促进大家学习英语，鼓励大家每周二用英语来交流。看到这个通知的时候，大家都觉得有意思，而又有难度。几个月过去后，这种交流慢慢显出效果。愿意练习英语的同事，在这一天尝试着在日

常工作交流中说英语，形成了一个良好的氛围。虽然有的时候要用汉语来补充，但那有什么关系，大家都在进步嘛。

在这里，我想建议大家要抽出时间去看一些汉语的小说、散文，动手写写汉字（手写，而不是用计算机），避免提笔忘字。如果汉语生疏了，实在说不过去啊。对于大多数人而言，我们所能真正体会到的一种语言的优美和深刻内涵的，我们所能被一部文艺作品的语言所感动的，还是我们的母语。

4.21 软件测试实例训练

计算机行业的一个重要特点就是有很强的实际操作的要求，做为其中之一的分支——软件测试也是这样。前面我们介绍了一些测试方面的理论，现在是做一个实例练习的时候了。

MSN很方便，实现了朋友之间方便、快捷地沟通，但不知道读者朋友有没有这种感觉： MSN7.5以及以前的版本，有一个不方便的地方，那就是MSN里的联系人一多，就不记得谁是谁了。因为MSN7.5以及以前的版本只能显示联系人自己确定的名称，而这个名称是随时可以改变的，所以如果联系人多了，加上他们的名称总是变来变去，就会使用户很糊涂。基于这种需求，我编写了一个小工具——MSN小助手，它可以让用户对MSN中的每位联系人增加一条备注，例如真实姓名，以方便用户及时辨别联系人的身份；还可以记录一些相关信息，例如联系方式、个人特点等。我们就以这个工具的测试为例，来应用我们刚学到的测试理论，锻炼我们的测试能力。

4.21.1 MSN小助手的软件需求规格说明书

为了方便测试工作的开展，我这里把MSN小助手的需求做了一个总结：

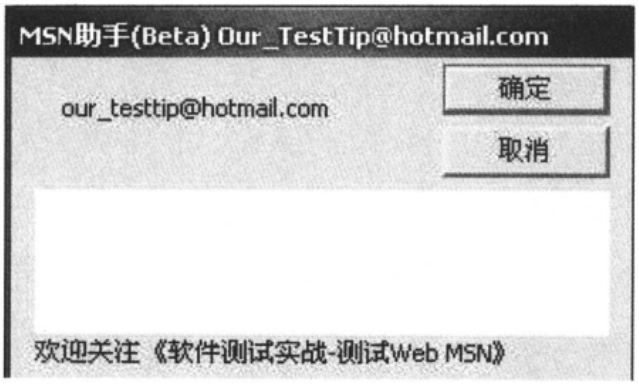
MSN小助手的需求说明

MSN小助手应当实现的功能是识别MSN中的联系人，允许用户给每位联系人添加一条备注，例如联系人的真实姓名、联系方式、特点等，这样用户在MSN上再次遇到这位联系人时，能及时、正确地辨别出联系人的身份。

当用户在MSN里添加了一位联系人A后，用户单击这位联系人，MSN小助手弹出一个对话框，显示联系人的邮件地址（因为联系人的邮件地址是惟一的，各不相同）。用户可以在这个对话框中输入一些信息，例如这位联系人的真实姓名、电话、住址或者其他一些需要记录的信息，等等。用户输入完毕后，单击“确定”按钮或者直接按回车键，对话框消失，信息被存储下来。用户可输入的信息的长度是100，用户可以输入不超过100个的中文字符。

用户“软件测试初学者”有一个联系人“齐月”，用户单击“齐月”后，出现下图的场景：

当用户查看联系人列表时，只要用户单击联系人A，MSN小助手就会弹出一个对话框，显示用户已经输入的信息，方便用户快速地识别联系人A的身份。MSN小助手的界面设计如下图所示：



MSN小助手应当支持所有的MSN版本，其中MSN7.0和MSN7.5更重要一些。

MSN小助手应当支持所有常用的Windows操作系统（Win98以及以后），例如Win98，Win2000，XP，XPsp2。

MSN小助手应当安装方便，用户只需要手动启动一次，以后这个软件就在操作系统启动的时候被自动加载。不能要求用户为MSN小助手而安装.Net Framework。用户可以方便地卸载这个软件。

MSN小助手应当使用方便，因为如果不方便，即使是免费软件，也不会有用户。

不要认为这个软件太简单了，没什么意思。一般来说，能做为软件测试教学案例的软件，都是简单的软件，目的是通过这个案例来运用学到的理论知识，锻炼实际测试能力，复杂的案例不利于交流。与其在一个复杂的案例中绕来绕去，不如通过一个简单的练习来训练我们的测试能力。

当我们做为一个新手来面对这个练习的时候，首先考虑的问题当然是，这个软件我都要测试哪些方面？下面我们就来数一数：

（1）首先要测试MSN小助手的功能，这是肯定的。例如能不能正确实现让用户为每一位联系人增加备注？能不能正确显示以前添加的备注？这

就是功能测试。

（2）看看使用它方不方便。例如，它是否和MSN协作良好？是不是能实现在操作系统启动时的自动加载？这是易用性测试。

（3）看看它运行的速度如何。观察它会不会越来越慢，因为用户一般会在上班期间持续地保持MSN在线，MSN小助手也会始终相伴，如果MSN小助手在长时间运行时出现越来越慢的情况，势必会影响用户的体验。这是性能测试。

（4）因为有对话框，所以要看看界面设计是否合适。这是UI测试。

（5）兼容性是需要考虑的，因为需求规格说明书中已经明确了，要MSN小助手与各种MSN的版本、各种操作系统保持兼容。这是兼容性测试。

好，我们想到了这么多，一个测试的粗略的框架就搭建起来了。下面我们一项一项地讨论，具体到测试用例的层次上。如果你要问我是怎么想到这5点的，我的回答是：

（1）这些测试点主要来自于MSN小助手的需求规格说明书。需求规格说明书是软件测试的标准，请仔细阅读它。

（2）还有一部分来自于一些软件的“公理”。也就是说，即使软件规格说明书里没有明确提出来，有一些“公理”仍是MSN小助手应当遵循的，例如要方便客户使用，界面设计要符合用户的规范，要允许客户长时间使用它，等等。

软件测试是一个渐进的过程，随着我们考虑的深入，我们可能还会发现在这5方面之外需要测试的，没有关系，随时可以添加。

这个小工具是我自己编写的，没有经过正式的测试，所以一起来试试吧，你肯定能找到一些bug☺。你可以从我的个人网站http://www.TestTip.cn上下载MSN小助手或者来信索取，我的E-mail是：Our_Test@hotmail.com。

4.21.2 功能测试

在考虑功能测试的时候，首先我们不管任何测试理论，对MSN小助手做两个最基本的确认，就是两个按钮对应的功能：

- 验证用户可以给一位MSN联系人添加一条备注（确定按钮）
- 验证用户可以取消对备注的修改（取消按钮）

现在我们来尝试着应用前面学过的测试理论。你可能看到了需求规格说明书中100这个数字，似乎有所考虑。看到这个数字，你是否想到了边界值测试法？我们再想一想，除了100这个数字这个边界值，还有没有其他的边界值呢？0也是一个边界值，你想到了吗？

现在，我们应用边界值的理论来设计一些测试用例：

- 验证用户可以存储长度为0的备注
- 验证用户可以存储长度为1的备注
- 验证用户可以存储长度为99的备注
- 验证用户可以存储长度为100的备注
- 验证用户不能完整存储长度为101的备注，第101个字符将不能被存储

同时，我们也应当选择其他一些有代表性的长度来做测试：

- 验证用户可以存储长度为20的备注
- 验证用户可以存储长度为56的备注
- 验证用户可以存储长度为84的备注
- 验证用户不能完整存储长度为1000的备注，第100个字符后的内容将被忽略
- 验证用户不能完整存储长度为10000的备注，第100个字符后的内容将被忽略

现在该轮到等价类理论粉墨登场了。需求规格说明书里当然不可能明示什么是我们需要考虑的等价类，需要我们自己去寻找。我们需要寻找多种等价类，就是说要找多种划分的方法，而不能局限于一种。打个比方，我们要把朋友进行分类的时候，可以按照来自的省份分，按照性别来分，按照所学的专业来分，按照性格来分，按照工作的城市来分，等等。

我们来看输入的字符的划分。一般来说，有代表性的字符有以下几类：

（1）单字节字符，英文是代表。英语在中国也是常见的语言，MSN小助手应当支持英语。

（2）双字节字符，代表语言是中文、日文、韩文。

（3）从右向左排序的字符，代表语言是阿拉伯文。虽然阿拉伯文不大常见，但在测试的过程考虑到总没有错。

（4）四字节的字符。咱们中国有一个GB18030的规范，囊括了中华民族的各种语言文字，字长从1字节到4字节不等。我们这个小软件MSN小助手暂不需要支持GB18030。对GB18030感兴趣的朋友可以到网上查询相关资料。

以字符的长度为等价类划分的标准，我们得到以下的测试用例：

- 验证用户可以存储英文的备注
- 验证用户可以存储日文的备注
- 验证用户可以存储中文的备注
- 验证用户可以存储多语言混合的备注

因为讨论到了字符，我们可以对一些特殊字符做验证：

- 验证备注中可以包含英文标点符号
- 验证备注中可以包含中文的各种标点符号
- 验证备注中可以包含“/”（因为“/”在计算机语言中有转义的作用，所以要做一个验证）

- 验证备注中可以包含“=”
- 验证备注中可以包含“&”
- 验证备注中可以包含“||”
- 验证备注中可以包含“；”
- 验证备注中可以包含“@”

还有没有可以进行分类的等价类呢？粗略一看，似乎只有按字符划分这一种了。

下面我们来分析一下MSN小助手的工作流程：

（1）用户单击；

（2）识别出邮件地址；

（3）用户输入；

（4）存储；

（5）再显示。

其中“识别邮件地址”这一项，使用过MSN的用户可能知道，MSN的邮件地址可以有这么几类：

（1）hotMail.com；

（2）msn.com；

（3）live.com；

（4）其他的合法邮件地址，例如163.com，sina.com。

从上面的分类可以看出，邮件地址也可以用来做等价类的划分：

- 验证MSN小助手可以识别出邮件地址后缀为hotMail.com的联系人
- 验证MSN小助手可以识别出邮件地址后缀为msn.com的联系人
- 验证MSN小助手可以识别出邮件地址后缀为live.com的联系人
- 验证MSN小助手可以识别出邮件地址后缀为163.com的联系人
- 验证MSN小助手可以识别出邮件地址后缀为sina.com.cn的联系人（邮件地址后缀为3段式的）

我们猜测开发人员在实现“识别”这个功能的时候，可能以“@”为标识符，对于邮件地址具体是什么并不关心，但做为测试人员，我们并不知道具体的实现方法，所以，如果再考虑细致一些，我们还可以考虑MSN小助手对不同的邮件地址格式的支持，得到下面的这些用例：

- 验证MSN小助手可以识别出邮件地址为全数字的联系人（以hotMail.com为例）
- 验证MSN小助手可以识别出邮件地址为全字母的联系人（以hotMail.com为例）
- 验证MSN小助手可以识别出邮件地址中有其他字符的联系人，例如下划线、中划线等（以hotMail.com为例）
- 验证MSN小助手可以识别出邮件地址为数字、字母、其他字符混合的联系人（以hotMail.com为例）

在讨论功能测试的最后，我们选择一个前面已经列出的测试用例，套用模版，把它详细地写出来，供大家参考：

测试用例：验证备注中可以包含“@”

优先级：2

准备步骤：

1. 登录MSN。
2. 运行MSN小助手。

测试步骤：

1. 打开MSN联系人列表。
2. 单击联系人Nice Man。
3. MSN小助手弹出对话框。
4. 用户输入“他是张三，他另一个邮件地址是hello@163.com”。
5. 点击确定，对话框消失。

预期结果：

1．当用户在MSN中再次点击Nice Man，在MSN小助手弹出的对话框中的信息是“他是张三，他另一个邮件地址是 hello@163.com”。

4.21.3 易用性测试

在考虑易用性测试的时候，我们要完全站在用户的立场去考虑问题，考虑这款软件用起来方不方便呢？

- 验证用户单击MSN中的一位联系人后，对话框弹出并处于桌面的最上层，用户可以直接输入信息
- 验证用户输入完备注后，可以直接以回车的方式来存储
- 验证用户只要在保存前，任何时候都可以取消修改
- 验证MSN小助手不会影响用户使用MSN
- 验证MSN小助手不会影响用户对其他软件的使用
- 验证如果用户没有登录MSN，MSN小助手不会弹出对话框
- 验证如果用户登录了MSN但没有单击联系人，MSN小助手不会弹出对话框

如果用户输入的备注信息较多，文本框中显示不全，这个时候如果有滚动条的话就会方便用户，所以：

- 验证当文本框无法把备注信息显示完整的时候，文本框中会出现垂直方向的滚动条
- 验证在有滚动条的情况下，如果用户删减备注信息直到文本框可以完整显示，滚动条会自动消失

以下几条是关于安装/卸载方面，你可以把它们单独列入安装测试的范畴。这里考虑到它们是安装测试和易用性测试的交集，暂列在这里：

- 验证用户只需要手动启动一次MSN小助手，以后它能在操作系统启动时自动加载

- 验证用户可以方便地安装MSN小助手，并且没有其他连带的安装要求（例如.Net Framework）
- 验证用户可以方便地更新MSN小助手的版本
- 验证用户可以方便地卸载MSN小助手

4.21.4 性能测试

MSN小助手只是一个单机版的小工具，粗看起来似乎没有做性能测试的必要。因为一谈到性能测试，似乎都是与C/S结构、大软件、多用户等相关。做性能测试的目的是什么？是保证当用户使用软件的时候对性能满意。MSN小助手再小，也不能在性能上有问题，例如，如果只能连续运行几个小时，连日常的使用场景——8小时都坚持不下来，谁又会再用它呢？所以，我们要在性能测试上花点时间。

首先，我们验证一下处理速度：

- 验证MSN小助手添加一条备注的处理速度足够快，能够满足用户的要求
- 验证MSN小助手读取和显示一条备注的处理速度足够快，能够满足用户的要求

如果用户连续地进行新增或者查看备注信息，处理速度是不是也能满足要求呢？

- 验证在用户连续新增20条备注信息的情况下，MSN小助手的处理速度能满足用户的要求
- 验证在用户连续查看20条备注信息的情况下，MSN小助手的处理速度能满足用户的要求

在做验证MSN小助手长时间运行的性能的测试之前，我们需要选择几个有代表性的使用场景。

（1）办公室职员A，他在整个上班时间都使用MSN和MSN小助手，一

天中（8小时上班时间）需要使用MSN小助手10次，其中2次为新增备注，8次为查看备注信息。

（2）人力资源专员B，她在上班时间需要频繁使用MSN和MSN小助手，一天中（8小时上班）需要使用MSN小助手100次，其中15次为新增备注，85次为查看备注信息。

（3）技术人员C（例如测试工程师），他在上班时间会打开MSN和MSN小助手，但使用频率不高。每天使用MSN小助手5次，1次为新增备注，4次为查看备注信息。同时，他下班后经常不关机，一般一周关机一次，如下表所示。

从上面3个例子中我们可以归纳出三个性能测试的场景，如下表所示：

性能测试场景1	
连续运行时间	8小时
新增备注	2次
查看备注	8次
性能目标	CPU占用率、内存使用量没有增加，响应速度满足用户的要求

性能测试场景2	
连续运行时间	8小时
新增备注	15次
查看备注	85次
性能目标	CPU占用率、内存使用量没有增加，响应速度满足用户的要求

性能测试场景3	
连续运行时间	24×5小时
新增备注	1次 / 24小时
查看备注	4次 / 24小时
性能目标	CPU占用率、内存使用量没有增加，响应速度满足用户的要求

我们的3个性能测试用例是：

- 验证MSN小助手在性能测试场景1达到预期性能目标
- 验证MSN小助手在性能测试场景2达到预期性能目标
- 验证MSN小助手在性能测试场景3达到预期性能目标

4.21.5 UI测试

因为MSN小助手只有一个界面，对话框，所以我们的UI测试就集中在这个界面中。

- 验证对话框的标题栏的内容显示正确
- 验证对话框中的字体、字体颜色、字体大小符合用户的使用习惯
- 验证用户不可以更改对话框的尺寸大小
- 验证对话框中的控件摆放位置以及它们的大小是合适的

4.21.6 兼容性测试

在兼容性方面，MSN小助手的需求规格说明书中已经有了相应的规定，我们把它们列出来就可以：

- 验证MSN小助手支持MSN7.5
- 验证MSN小助手支持MSN7.0
- 验证MSN小助手支持Win98

- 验证MSN小助手支持Win2000
- 验证MSN小助手支持Windows XP
- 验证MSN小助手支持Windows XP SP2

4.21.7 尝试从开发方面获得更多的信息

讨论到这里，我们已经在测试方面下了一些功夫，现在要做的就是提高了。从做这个练习（MSN小助手的测试）开始，我们一直在测试这条线上，提都没有提到过开发那一边。其实，如果我们能知道开发人员的具体实现方法，我们还可能新增一些测试用例。下面是一些“开发内幕”：

（1）MSN小助手会在硬盘的C盘的根目录下存储备注信息，格式是文本文件。

（2）MSN小助手会拦截操作系统中所有的获得焦点（Focus）的消息。

（3）MSN小助手是这样识别MSN中的联系人的：通过操作系统的Accessibility机制读取当前对象的辅助信息，如果读出的辅助信息中有邮件地址，就把它判断为MSN中的一个联系人（这并不能实现精准判断）。

你可以试试，根据这些信息再编写出一些测试用例，以发现更多的bug！

至于如何得到更多的开发方面的信息，方法有很多种，例如阅读设计文档、阅读源代码等。另外，与开发人员的交流往往也能获得有用的信息，有些信息甚至就来自于餐桌上，所以保持与开发人员良好的关系会促进测试工作☺。

05

第5章 典型的测试工程师笔试题

现在基本上所有的公司在招聘测试工程师的时候都有笔试的要求，根据我自己的工作经验以及从网上找到的一些相关资料，把一些有代表性的测试工程师的笔试题列在下面，同时做了一些分析，并给出了参考答案，供大家参考。

5.1 编程

虽然有一部分测试职位并没有编码能力的要求，但是越来越多的公司在对测试工程师的候选人进行考察时加入了编码能力的测试。而且，中级以及中级以上的测试工程师的职位对编码能力都是有要求的，希望大家有所准备。在这一节中，参考答案中的代码都是C#格式。

5.1.1 两数相除

题目：请编写一个函数，这个函数有两个实型的入口参数A和B，函数的返回值是A除以B。

分析：这道编程题是很简单的，它惟一需要注意的地方是要做一个分母是否为0的检查。

参考答案：

```
double Div(double A, double B)
{
    double result = 0 ;
        if (B == 0) result = 0 ;
        else
            result = A / B;

        return result ;
}
```

5.1.2 最大公约数

题目：请编写一个函数，求两个数的最大公约数。

分析：这道题是考察基本逻辑能力。你不一定要给出一个最佳的算法，但逻辑一定是要对的。我遇到过几位候选人，他们看到这道题以后就绞尽

脑汁地想，是不是有一个什么系统函数来求最大公约数？这是一种误解。用人单位是想看看你的编码能力，而不是让你做一个系统调用。

参考答案 1：

```
int GreatestDivisor(int a, int b)
    {
        int min= 1 ;
        int result = 1 ;
        if ((a<=0) || (b<=0))
        {
            result = 0 ;
            return result ;
        }

        if (a < b) min = a ;
        else
            min = b ;

        for (int i=min; i>=1; i--)
        {
            if (((a % i)==0) && ((b % i)==0))
            {
                result = i ;
                break ;
            }
        }
```

```
        return result ;


    }
```

参考答案 2：

```
int GreatestDivisor2(int a, int b)
    {
        int temp ;
        int result = 0 ;

        if ((a<=0) || (b<=0))
        {
            result = 0 ;
            return result ;
        }

        if (a < b)
        {
            temp = a ;
            a = b ;
            b = temp ;
        }

        while ((a % b)!=0)
        {
            temp = a % b ;
            a = b ;
```

```
                b = temp ;
            }

            result = b ;
            return result ;

        }
    }
```

5.1.3 排序

题目：有一个数组，数据类型是整数型，请写一段程序，把这个数组内的数做降序排列。

分析：排序是数据结构的基础知识，请掌握几种排序方法。排序是笔试中出现频率很高的题。

参考答案：

```
void sort(int[] VarArray)
    {
        int length = 0 ;
        int i, j, temp ;

        int[] MyArray = VarArray;

        //如果数组不存在，直接返回
        if (MyArray == null) return ;

        length = MyArray.Length ;
        //如果数组的长度小于等于1，不需要排序，直接返回
```

```
        if (length<=1) return ;

        for (i=0; i<length-1; i++)
        {
            //在第二层循环中，每执行一遍，都能找到一个最小
            //的数，把它置换到数组的尾部
            for (j=0; j<length-1-i; j++)
            {
                if (MyArray[j] < MyArray[j+1])
                {
                    //下面的三行代码只做一件事：把两个数
                    //对换一下
                    temp = MyArray[j] ;
                    MyArray[j] = MyArray[j+1] ;
                    MyArray[j+1] = temp ;
                }
            }
        }

    }
```

5.1.4 递归

题目：请用递归算法计算 N 的阶乘。

分析：递归是数据结构中比较难理解的部分，递归函数会调用自身，我们通过这道题来复习一下。

参考答案：

```
int Recursion(int n)
    {
        if (n<0) return 0 ;
        if (n==0) return 1;
        else
            return n*Recursion(n-1) ;
    }
```

5.1.5 程序阅读

1. 基本语法

下面是一段程序：

```
int exampleFunction(int a, int b)
    {
        int temp = 0 ;
        int result = 0 ;
        bool bFlag = false ;

        switch (a)
        {
            case 1:
                temp = -1 ;
                break ;
            case 2:
                temp = 5 ;
                break ;
```

```
            case 3:
                temp = 28 ;
                break ;
            default:
                temp = 0 ;
                break ;

        }

        if (b>0)
        {
            bFlag = true ;
        }
        else
        {
            bFlag = false ;
        }

        if ((temp==0) && !bFlag) return result ;

        while (bFlag)
        {
            result = result + b + temp ;
            if (result > 50) bFlag = false ;
            if (result < 0)
            {
```

```
                result = a + b + temp ;
                return result ;
            }
        }

    return result ;


}
```

题目：当入口参数 a=10、b=20 时，函数的返回值是多少？当 a = 3、b= -1 时呢？

分析：这类的程序阅读题属于比较简单的类型，考察的是候选人的基本逻辑能力。如果你觉得比较绕，就用笔一步一步写出中间过程，直到最后得出结果。这也是做程序阅读题的一个良好习惯。

参考答案：当入口参数a=10、b=20时，函数的返回值是60。当入口参数a=3、b=−1时，函数的返回值是0。

2. 指针

链表Student有4个节点：A、B、C和D，其中A为头节点，它们的关系是：

```
A.next = B ;
B.next = C ;
C.next = A ;
D.next = null 。
```

有一个程序，可以求链表的长度：

```
int LinkLength（StudentNode *stu）
{
    StudentNode * StudentLink;
```

```
        StudentLink = stu ;
        if （StudentLink = = null）return 0 ;

        int length = 0 ;
        while （StudentLink.next!= null）
        {
            Length ++ ;
            StudentLink = StudentLink.next ;
        }
        Return length ;

    }
```

题目：链表 Student 通过 LinkLength 求出的长度是多少？

分析：做指针的题目，我们要小心，看清楚它们的每一步指向。另外，我们还要克服我们自身的一个默认答案就是一定有一个输出，其实并不总是这样。这道题是个死循环，因为C的next并没有指向D，而是指向A。

参考答案：死循环

3．重载

```
class Picture
{
    void Draw()
    {
        cout << “Class Picture” << endl ;
    }
}
```

```
class BMPPicture
{
    void Draw()
    {
        count << "Class BMPPicture" << endl ;
    }
}

int main()
{
    Picture *pic ;
    BMPPicture bmp ;
    Pic = &bmp ;
    pic->draw();

    return 0 ;
}
```

题目：这段程序的输出结果是什么？

分析：pic是一个Picture类型的指针，尽管它被赋予了BMPPicture类型的对象bmp，但是因为在编译时不会指向具体的对象，所以pic->draw()还是会被认为是对Picture的draw的调用。

参考答案：

输出结果是Class Picture。

5.2 数据库

5.2.1 基础的SQL语句

分析：最基础的SQL是指实现增、删、改和查询的SQL语句，这是数据库的基础知识，是一定要掌握的。

在数据库中有一张候选人表（Candidate），它的定义如下：

姓名(Name，主键1)

性别 (Sex)

身份证号（ID，主键2）

主要经历（Resume）

应聘工作（ExpectedJob）

面试评价（Comments）

题目：请写出下列 SQL 语句

（1）列出所有候选人的姓名和身份证号。

参考答案：Select Name, ID from Candidate

（2）增加一条记录：

姓名：张三

性别：男

身份证号：123456789123456789

主要经历：2003-2007年就读大学，专业为计算机

应聘工作：1（1为代号，意指软件测试）

面试评价：计算机知识扎实，测试能力不错，可以试用

参考答案：Insert into Candidate（‘张三’，‘男’，‘123456789123456789’，‘2003-2007年就读大学，专业为计算机’，‘1’，‘计算机知识扎实，测试能力不错，可以试用’）

（3）删除主要经历和面试评价都为空的记录。

参考答案： Delete from Candidate where Resume='' and Comments=''

（4）更新姓名为李四、身份证号为123456123456123456的记录的面试评价为：列入考虑范围。

参考答案： Update Candidate set Comments='列入考虑范围' where Name='李四' and ID=' 123456123456123456'

（5）列出所有姓王的候选人的记录。

参考答案： Select * from Candidate where Name like '王*'

5.2.2 复杂一些的SQL语句

图书馆有一套图书管理系统，有一个数据库LibraryManagement，其中有4张表：读者信息表，读者分类表，读者借阅表，书籍信息表。各表内容列出如下：

读者信息表（Reader）：

序号（ID）
姓名（Name）
性别（Sex）
类别（Category）
联系地址 （Address）
电话 （PhoneNumber）
电子邮件 （EMail）

读者分类表（ReaderCategory）：

序号（ID）
类别（Category）
限量（Limit）

备注：例如，普通读者一次只能借3本书，1个月内需要归还；会员读者可以借5本，2个月内需要归还。

读者借阅表（BorrowInfo）：

流水号（ID）

读者编号（ReaderID）

时间（Time）

所借书的编号（BookID）

是否归还（Returned）

书籍信息表（Book）：

编号（ID）

书名（Name）

出版社（PublishHouse）

作者（Author）

题目 1：请从读者表中找出读者编号在 10000 到 20000 之间的记录。

分析：这道题主要是考察候选人对条件语句的使用，有多种实现方法。

参考答案1：Select * from Reader where ID BETWEEN 10000 AND 20000

参考答案2：Select * from Reader where ID >= 10000 and ID <= 20000

题目 2：显示读者 10001 所借的书籍的详细信息。

分析：这道题是要考察候选人对两个表中读取数据的能力，主要是要根据书的ID把具体信息从书籍信息表中读出来。

参考答案：Select Br.ReaderID, Br.Time, Bo.Name, Bo.PublishHouse, Bo.Author from BorrowInfo Br, Book Bo where Br.ReaderID=10001 and Br.BookID = Bo.ID

题目 3：显示编号为 00001，00002，00003 的读者的详细信息，要求用 IN。

参考答案： Select * from Reader where ID IN (00001, 00002, 00003)

题目 4：查询读者 10001 总共借阅了多少本书。

分析： 这是考察读者对SQL函数的使用，包括AVG（平均），COUNT（统计个数），MAX（最大值），MIN（最小值），SUM（总计）。

参考答案： Select COUNT（*） from BorrowInfo where ReaderID=10001

题目 5：显示已经借过书的读者的编号和借阅量。

分析： 这道题需要统计多个读者的情况，可以使用GROUP BY。

参考答案： Select ReaderID, COUNT（*） from BorrowInfo GROUP BY ReaderID

题目 6：显示已经借阅 20 本书以上的读者的编号和借阅量。

分析： 这个问题比问题5更进一步，要求只显示其结果的一部分数据。在GROUP BY的语句中，我们使用HAVING来做进一步的条件限制。

参考答案： Select ReaderID, COUNT （*） from BorrowInfo GROUP BY ReaderID HAVING COUNT （*）> 20

题目 7：把借阅表中的信息按照读者的 ID 做升序排列显示。

分析： 这是要考察对ORDER BY的使用，关键字DESC（Descend）是表示降序，ASC（Ascend）表示升序。

参考答案： Select * from BorrowInfo ORDER BY ReaderID ASC

题目 8：请查询读者 10001 还能借阅几本书。

分析： 要得到一位读者还能借阅几本书的信息，我们需要得到两个数据：一个是这位读者有借几本书的权利，另一个是他已经借了几本书（尚未归还的书），我们先分别写出得到这些数据的SQL语句。

读者可以借几本书： Select Rc.Limit from Reader Re，ReaderCategory Rc where Re.ID = 10001 and Re.Categery = Rc.ID

读者已经借了几本书：Select COUNT(*) from BorrowInfo where Reader = 10001 and Returned = 0

参考答案：

Select availeCount=（Select Rc.Limit from Reader Re,ReaderCategory Rc where Re.ID = 10001 and Re.Category = Rc.ID） — （Select COUNT（*） from BorrowInfo where ReaderID = 10001 and Returned = 0）

5.3 测试理论

题目 1：一个 bug 的描述中通常都包括了哪些内容？

分析：这道题的用意是检查候选人是否掌握了基本的测试知识。因为测试工程师的工作成果就是以bug来体现，所以候选人对bug包括哪几部分是应当很清楚的。

参考答案：一个bug通常包含以下内容：

（1）概要；

（2）重现步骤，这是最重要的；

（3）产品版本号；

（4）重要环境参数（例如操作系统）；

（5）优先级；

（6）严重性；

（7）状态；

（8）发现者和接受者（发现者一般都是测试人员，接受者一般都是开发人员）；

（9）原因分析；

（10）如果有图片的话，附上图片。

题目 2：Bug 有几种状态？

分析：Bug的状态转换在实际测试工作中很重要，bug的生命周期正是在状态转换中实现的。因为有些状态在没有实际工作经验的时候是很难记住的，所以这还可以考察一下候选人的实际测试经验，没有工作经验的朋友可能要强记一下。

参考答案：Bug的常见状态有：

（1）打开；

（2）修正（即已经被开发人员修正了，等待验证）；

（3）关闭；

（4）不必修正（没有必要做修正）；

（5）重复的bug（已经有描述同一问题的bug）；

（6）以后再修正（例如，不重要的，或者时间已经来不及了）；

（7）外部的bug（例如，在应用程序中发现一个bug，却发现是操作系统的一个bug引起的，因为无法去修正操作系统的bug，所以可以把这个bug标识为外部的bug，意思是说不是本项目所能解决的）；

（8）假问题（测试人员的工作失误）；

（9）其他（无法归类的）。

题目 3：Bug 的优先级和严重性之间的区别是什么？

分析：在实际的工作中，有些项目只是简单地把bug分级，却没有区分优先级和严重性。回答这类问题的时候，可以尝试举例，这样更容易说明清楚。

参考答案：Bug的优先级是指一个bug需要及时解决的程度，而严重性是指一个bug对产品的影响程度。一般来说，优先级与严重性是成比例的，即严重性越高的bug，优先级也越高。但是，它们并不总是一一对应，例如，一个界面上的bug，对于产品的功能来说并没有什么大的影响，严重性不高，但是因为这个bug很容易被客户发现，所以优先级很高。

题目 4：你认为做好软件测试用例设计的关键是什么？

分析：这类的题目有一定的主观性，允许候选人根据自己的理解来回答。就我看来，测试用例的目标是更高的测试覆盖率，所以测试用例设计的关键应当是提高测试覆盖率。同时，回答这个问题的时候我们还可以做一个延伸，去讨论一下如何能实现这个“关键”，这样可以让回答更圆满一些。

参考答案：我个人认为做好软件测试用例设计的关键是实现高的测试覆盖率。如何实现高的测试覆盖率呢？可以从以下几个方面去着手：

（1）熟悉需求规格说明书；

（2）熟练使用多种测试方法；

（3）多与同事讨论；

（4）从类似项目中做借鉴；

（5）做好评审。

题目 5：软件测试结束的标准是什么？

分析：虽然理论上说我们测试工程师无法找出一款软件产品的所有bug，但是软件测试毕竟有结束的时候。在考虑何时退出测试的时候，我们既要考虑软件产品满足需求规格说明书的定义的程度，也要考虑市场的因素。

参考答案：

软件测试结束的标准有两方面：

（1）软件产品是否符合需求规格说明书的要求。

（2）市场的因素。这也是测试人员容易忽略的一个地方。测试工程师都有一种“完美”的倾向，总是希望软件尽善尽美，希望所有的bug都能被修正。然而，商场就是战场，产品需要在预定的最佳时机上市，否则，如果错过了好的时机，让对手抢了先机，就是一个大的失策。

我们需要在软件产品的质量和市场要求之间取一个平衡，来确定什么

时候结束测试。另外，还有一种特殊情况，如果在软件研发过程中，软件质量出现重大问题，以至于没有必要再继续研发下去，测试也可以在与各方协商后提前退出，这一般意味着软件项目要被中止。

题目6：什么是等价类的划分？请举例说明。

分析：等价类的划分是软件测试的一个重要的理论基础，请务必掌握。下面的参考答案是我的理解，你也可以参考相关的软件测试教材，也可以以理论联系实际的方式来回答。

参考答案：等价类的划分是软件测试的基本理论，它是指把测试集合按照一定的标准划分为几类，测试的时候从每一类中选择一个或者几个有代表性的数据进行测试。例如，计算个人所得税软件中，个人工资额度是一个很重要的因素，不同的额度对应不同的税率。假设，2000~3000元之间执行税率A，3000~4000元之间执行税率B，因为实际的工资额度千变万化，没有必要一一做测试，我们可以把它划分为几个等价类：

（1）2000元以下；

（2）2000~3000元；

（3）3000~4000元；

（4）4000元以上；

（5）负数；

（6）非数值型。

然后从每个等价类中选择几个数据来做测试就可以了。

题目7：什么是边界值测试法？请举例说明。

分析：边界值测试法也是软件测试的一个基本理论，请务必掌握。理论是简单的，配上实例更容易清楚说明问题。

参考答案：我认为，在边界值测试法中，每一个条件的边界是最容易出现错误的地方。如果N是一个集合的边界的话，那么根据边界值测试法至少需要测试三种情况：N-1、N、N+1。例如，在一款法院管理软件中，年

龄是判断犯罪嫌疑人是否承担刑事责任的一个条件，其中16岁是一个边界值。那么做测试的时候，我们至少测试3个数：15、16、17。

5.4 测试实践

题目 1：请给出一个完整的测试用例

分析：这道题实际是想检查候选人是否有测试工作经验，以及是否了解软件的基础知识。这道题本身并不难，对于想进入软件测试行业的朋友来说也应当掌握。在软件测试用例中，最重要的是要给出预期结果，在这一点上的差别能够考察出候选人有没有学这测试。

软件测试用例是软件测试的基础性文档，是软件测试工程师的测试思想的重要体现，它通常包含以下部分：

（1）标题（概要）；

（2）模块名；

（3）优先级；

（4）测试环境要求；

（5）测试步骤；

（6）预期结果。

最简单的测试用例包括以下部分：

（1）标题；

（2）测试步骤；

（3）预期结果。

在笔试中最好是写一个自己写过的测试用例，不管是在工作中还是在学习中写过的，而尽量少地去编撰，因为面试官很可能会在面试中与你讨论你的笔试题。

参考答案：

这是我曾编写过的一个测试用例。软件产品是MSN，功能模块是聊天

模块。

标题（概要）：验证MSN能够发送中文消息

模块名：MSN聊天功能

优先级：0

准备步骤：

1. 准备两台计算机，操作系统都是XPSP2，安装版本号为7.0的MSN；
2. 准备两个MSN帐号：A、B、A和B互相为联系人。

测试步骤：

1. A和B分别在不同的计算机上登录MSN；
2. A打开一个对B的聊天窗口；
3. A输入一句中文：“你好，现在忙吗？”；
4. A点击发送。

预期结果：

1. B接收到A发送过来的消息：“你好，现在忙吗？”，消息显示正确。

题目 2：请给出一个完整的 bug 报告

分析：Bug是测试工程师测试结果的重要体现，如果不能清楚描述bug，不懂得bug中应当包含的信息，这很难让面试官信服你有做软件测试的能力。Bug报告一般包含以下部分：

（1）项目名称；

（2）模块名称；

（3）标题（概要）；

（4）优先级；

（5）严重级别；

（6）类别；

（7）产品版本；

（8）里程碑；

（9）重现步骤，包括问题所在和预期结果。

其中“重现步骤”最重要，是不可缺少的信息。

参考答案：

这个bug是我以前提交过的一个bug，产品是学生管理系统，模块是成绩统计。

项目名称：学生管理系统

模块名称：成绩统计

标题（概要）：无法输入大于100分的成绩

优先级：1

严重级别：1

类别：逻辑错误

产品版本：1.0

里程碑：第一阶段（M1）

重现步骤：

1. 进入学生管理系统；
2. 进入成绩统计界面；
3. 点击“新增学生成绩”；
4. 输入“赵建国、英语、110”，因为有的试卷是120分制或者150分制，所有这种输入是有可能的；
5. 点击“确定”。

问题：

1. 出现一个系统提示框：“成绩只能在0~100分之间，请重新输入！”，这个限制是不正确的。

预期结果：

2. 添加成功。

原因分析：限制条件设置不正确，没有考虑到一些特殊的考试分数。

题目 3：请描述测试计划的要点

分析：因为笔试的时间一般都是在1小时到1个半小时，公司一般不会要求候选人写一个完整的测试计划。但是为了考察候选人以前是否编写过测试计划，就会出这样一道题，让候选人写出测试计划的要点。所以，即使是软件测试的初学者，也应当对软件测试计划做一个了解。

参考答案：

测试计划有多种形式，但其中的主要内容都是一样的。一个测试计划应当包含以下几部分：

（1）软件产品简介；

（2）研发进度（包括测试进度）；

（3）人员安排；

（4）资源（硬件、软件）安排；

（5）测试策略和方法；

（6）测试范围和退出标准；

（7）可能的风险以及应对办法（可选）。

备注：对于候选人的测试计划的考察，还有一种方式，即给出具体的软件产品的简单描述，然后要求候选人写一个简单的测试计划。遇到这种情况的时候，我们可以按照上面的框架，填充进去实际内容，简明扼要即可。

题目 4：测试 MSN 的聊天机制。

注：本题仅限于文本形式的聊天，不必考虑音频、视频等。

分析：这道题主要是想考察候选人测试思维能力，看候选人能列出多少需要测试的情况。MSN是大家都熟悉的软件，聊天功能是核心功能。之所以增加了一个约束条件——文本形式，是想把题的范围缩小，以便于候

选人集中精力思考问题。回答这类题目的时候，一般是以思路的“广”取胜，就是看谁想得多，所以请详细列出自己的思路。

这里面还有几个重要的考察点是大家需要注意的：

（1）MSN的聊天一般都是在两人之间或者两人以上进行的，那么判断一条消息是否发送成功需要在接收方验证。例如，A发送消息出去，B是接收方，结果需要到B那里验证，这是需要明确写出来的。

（2）不要局限于两人聊天，MSN支持多人聊天。

（3）MSN有很多版本，不同版本之间的兼容性是需要考虑的。

（4）对不同语言的支持。

（5）如果还能考虑与Yahoo聊天工具的互联互通，那就更好了。

参考答案：

聊天功能是MSN的核心功能，仅文本形式的聊天就比较复杂。因为MSN的特殊性，验证结果需要到接收方验证，而不能局限在发送方，这是在测试过程中需要遵循的一个原则。对于MSN的聊天机制的测试，我会从以下几个方面去考虑：

（1）两人聊天

- 普通的聊天场景；
- 验证对一次发送消息的长度的限制；
- 双方处于不同状态下的聊天，例如忙碌、离开、在线、离线等；
- 多个聊天场景同时存在的情况，例如，A与B在聊天，A又开了一个窗口与C聊天；
- 使用方便性方面的测试；
- 聊天记录的保存。

（2）三人聊天

- 三人聊天的发起与退出；
- 三人在不同状态下的聊天。

（3）三人以上参与聊天

（4）不同版本之间的MSN的聊天功能的兼容

（5）对不同语言的支持

- 英文
- 中文
- 日文
- 中文GB18030字符
- 阿拉伯文
- 一条消息中包含多种语言

（6）与Yahoo聊天工具的互联互通

（7）测试MSN在聊天过程中对网络不通等特殊情况的检测能力和恢复能力

题目 5：测试自动售票机。

在北京火车站有一种自动售票机，是触摸屏式的，出售北京到天津的火车票。它只接受新版的面值为100的人民币，根据用户的选择打印出火车票，并找零。北京到天津的城际列车的票价因为时间的不同而存在3种票价：25、30、35，每人每次限买2张。

备注：此题的描述和自动售票机的实际情况可能略有出入。

分析：这道题很实际，也很有意思。即使不是笔试题，我们也可以把它当作训练思维能力的小游戏。面试官在出面试题的时候，一般都喜欢“以小见大”，就是希望试题简单，但又能通过这道题看出候选人的实际水平。

在考虑怎么测试之前，我们来看看自动售票机和什么相关：

（1）售票软件；

（2）数据库；

（3）网络状况；

（4）车票打印设备，打印纸；

（5）钞票识别设备；

（6）准备的找零（就是需要准备5元、10元、50元等面值的纸币）；

（7）电源。

参考答案：

我认为以下几个方面是测试的重点：

常见的购票场景

（1）买一张25元的车票；

（2）买一张30元的车票；

（3）买一张35元的车票；

（4）买同一车次两张车票的情况；

（5）买两张车票，但为不同车次，有两种情况，一种是两张车票同价，另一种是两张车票不同的价格；

（6）各种情况下的取消；

（7）对车票数量的限制；

（8）提示信息是否明确。

各种异常情况

（1）纸币不正确；

（2）各种情况下的无法连接数据库；

（3）打印机没墨；

（4）打印机缺纸；

（5）买两张票，打印纸只剩最后一张；

（6）缺5元的找零；

（7）缺10元的找零；

（8）缺50元的找零；

（9）缺2种找零；

（10）缺3种找零；

（11）确定交易后无票（被其他窗口售出）；

（12）交易过程中停电。

安全性测试

（1）在前一位顾客购票后，“黑客”随即使用售票机，看是否能偷票；

（2）“黑客”在最后交易前取消，看售票机是否还会出票；

（3）“黑客”是否能够利用浏览功能（显示车次和票数的查询）偷票；

（4）其他情况。

题目 6：三角形程序的测试用例。

有一个程序，能够判断用户输入的三个数是否能构成一个三角形。如果能构成三角形，程序还能判断出三角形的类型，例如等腰、等边、直角三角形等。

分析：这是一个很经典的问题，很多软件测试的教材中都引用了这个例子，它也多次出现在笔试的试卷当中。我曾经参加过微软公司的内部培训，这道题也在其中。它重点考察的是等价类的划分这个知识点。下面是我的几点提示：

（1）把关于三角形的数学知识都考虑到。例如，只有任意两边之和都大于第三边，才能构成三角形；例如，三角形的种类有普通三角形、等腰三角形、等边三角形、直角三角形、等腰直角三角形（锐角、直角和钝角三角形是另外一种分法），这些类型在测试中都是要考虑到的。

（2）边长不要局限在正整数，小数、零、负数等都要考虑。

（3）不要局限在功能测试上。有的题给出了题中描述的程序的界面，那么在功能测试之余，我们还要考虑界面测试、易用性测试、兼容性测试、性能测试等等。即使没有给出程序界面，我们也可以列出这些测试项。尽可能多地多想一些，考虑得周到一些，是我们测试人员应当做的事情。

在这里我做一个补充说明，在数学上并没有“普通三角形”一说，这里只是为了与其他特殊的三角形做一个区别，程序对此类三角形的提示为

“可以构成三角形”就可以了。

这道题有多种做法，下面我给出我的参考答案，供大家参考。

参考答案：

对于这个软件，重点是做功能测试，同时也要考虑界面测试、易用性测试、兼容性测试、性能测试等。

（1）功能测试

对非法输入的检查

- 有一边为非数值的字符
- 有两边为非数值的字符
- 三边皆为非数值的字符
- 有一边为0
- 有两边为0
- 三边皆为0
- 有一边为负值
- 有两边为负值
- 三边皆为负值
- 考虑一些可导致错误的符号，例如“/”、“&”、“|”等等

不能构成三角形的情况

要注意验证程序是否达到了“任何两边之和都应当大于第三边”中的“任何”的要求。以下的三边a、b、c要考虑正实数的情况，包括整数和小数。

- 两边之和等于第三边的情况
  - a+b = c
  - a+c = b
  - b+c= a

- 两边之和小于第三边的情况
  - a+b < c
  - a+c < b
  - b+c < a

能构成三角形的情况

以下的三边a、b、c要考虑正实数的情况，包括整数和小数。

- 验证对普通三角形的判断
- 验证对等腰三角形（非等边三角形）的判断
  - a=b
  - b=c
  - a=c
- 验证对等边三角形的判断
- 验证对直角三角形的判断
  - a、b为直角边
  - b、c为直角边
  - a、c为直角边
- 验证对等腰直角三角形的判断
  - a、b为直角边且a=b
  - b、c为直角边且b=c
  - a、c为直角边且a=c

（2）界面测试

- 验证控件的大小是否正确
- 验证控件的摆放是否正确
- 验证字体、颜色是否正确
- 验证界面在缩放的时候变化是否正确

（3）易用性测试

- 验证默认焦点是否正确
- 验证焦点顺序是否符合读者习惯
- 验证输入a、b、c完毕后，用户能否通过回车来直接触发判断功能

（4）兼容性测试

验证这个软件是否能运行在预期支持的各种操作系统上

（5）性能测试

验证程序的响应速度

（6）如果判断的结果以对话框的方式来显示，则还需要对这个信息对话框做测试

题目 7：测试一个杯子。

分析：这道题是我的一个朋友告诉我的，她在参加微软公司的面试的时候遇到了这道题。我不知道读者朋友是否有这样的感受，当我第一次听到这道题的时候心里在嘀咕着，这也能算是测试工程师的面试题？杯子不是软件啊，怎么测？无处下牙的感觉。其实，这道题有点智力题的感觉，但本质上还是测试题，招聘公司要考察的是你的思维的灵活性。这样的题没有标准的答案，面试官会从你的答案中对你的测试思维能力以及灵活性做一个判定。

即使你感到这道题再古怪，也不要放弃，一个字都不写是最糟糕的回答。我们来分析一下杯子：

（1）圈定测试的范围，结束无处下牙的局面。我们要想到，杯子总有一种类型，我们不可能去测试所有的杯子，即使题目里没有明确杯子的类型，我们也可以自己做一些假设。

（2）外观测试。外观设计方面，例如是否实用？是否美观等？

（3）功能测试。它是容器，是否能装水？能装固体吗？容量是多少？

（4）性能测试。会被摔碎吗？抗挤压吗？能被使用多少次？等等。

（5）易用性测试。装热水的时候容易烫手吗？方便旅行携带吗？等等。

下面是我给出的参考答案，还是那句话，这种类型的题的答案不是惟一的，欢迎你自己试试。类似的题目还有：如何测试一部电梯？如何测试一辆汽车？等等。

参考答案：

虽然杯子不是一款软件，但是我还是有信心去测试它。下面是我的测试思路：

（1）外观测试

- 形状是否方便使用？
- 是否美观？
- 杯子外壁的图案是否合适？
- 需要光泽吗？
- 高度、直径是否符合要求？
- 杯子的材质是否符合要求？

（2）功能测试

- 验证杯子能装水
- 验证杯子标注的容量是正确的
- 验证杯子能装开水
- 验证杯子能装冰水
- 验证杯子能装油（水以外的液体）
- 验证杯子能装固体（石头、沙子等）

（3）性能测试

- 杯子能被使用多少次？
- 容易摔碎吗？

- 抗挤压吗？

（4）易用性测试

- 杯子容易被洗干净吗？
- 是否方便用户拿着？
- 容易烫手吗？
- 有过滤茶叶的附加功能吗？

（5）国际化测试

- 杯子的形状、颜色是否冒犯了有些国家的禁忌？
- 杯子上的图案是否有政治、宗教方面的错误？

5.5 软件问题分析

测试工程师职位对应聘者的问题分析能力有比较高的要求，因为在搭建测试环境以及测试过程中随时都会遇到问题，这就需要测试工程师去分析和解决，所以出现“问题分析”一类的题是很正常的。

软件题目 1：在测试 Win Form 的 C/S 结构软件时，发现这个软件的客户端的运行速度很慢，你会采取哪些方法去检查和分析这个问题？

分析：虽然题目中已经定性了“问题”，其实软件运行速度慢下来不一定就是软件本身的原因，所以我们思考这类问题的时候可以从两方面去分析：

（1）是正常的情况，例如暂时的服务器繁忙；

（2）是异常的情况，例如内存泄露等。

参考答案：

我从以下几个方面去分析这个问题：

（1）是不是在执行什么特定的功能？

（2）是不是服务器处于繁忙的状态？例如，大量用户登录或者使用；

（3）是不是客户端的操作系统运行了大量的其他程序，导致这款软件不能及时得到响应或者内存分配不足？

（4）是不是网络出现故障？

（5）是不是连接数据库失败？

（6）是不是客户端出现内存泄露导致程序越来越慢？

软件题目 2：当你在搭建测试环境的时候，在 Window 下，点击“开始”，选择“关闭计算机”，显示出来的只有“注销”，而没有“重新启动”和“关机”，导致出现这种情况的可能原因是什么？

分析：如果你在实际中遇到过这个问题可能会有些思路。

参考答案：

可能的原因是：

（1）用户登录的帐号权限不足，没有“重新启动”和“关机”的权限；

（2）用户是通过远程登录到计算机上，默认情况下就不拥有“重新启动”和“关机”的权限；

（3）操作系统被病毒控制；

（4）未知的操作系统错误。

软件题目 3：浏览器无法打开指定的网站，请问可能的原因是什么？

分析：我们分析一下浏览器打开一个网页所关联的因素：

（1）操作系统；

（2）用户的帐号；

（3）浏览器；

（4）防火墙；

（5）网络状况；

（6）网关；

（7）域名解析；

（8）网站的权限设置；

（9）网站的状态。

我们分析问题的时候就可以根据上面的内容多做一些考虑。

参考答案：

我认为可能的原因是：

（1）操作系统方面：操作系统是否出现紊乱？

（2）用户的帐号方面：用户帐号是否具有上网的权限？

（3）浏览器方面：浏览器是否正常运行？浏览器的设置是否正确？

（4）防火墙方面：防火墙是否允许浏览器的运行？

（5）网络状况：网络是否畅通？带宽是否足够？

（6）网关方面：网关是否工作正常？

（7）域名解析方面：是否存在这个网站的域名解析记录？解析是否正确？

（8）网站的权限设置方面：网站是否拒绝匿名访问？网站是否拒绝一定的IP范围？

（9）网站的状态方面：网站是否存在？网站是否启动？该网页是否存在？该网页是否支持用户所使用的浏览器？

5.6 测试工作经验

题目 1：如何维持测试人员与开发人员良好的人际关系？

分析：因为在实际工作中测试人员与开发人员会有很多接触，而且客观地讲，测试人员需要给开发人员“挑刺儿”，所以测试人员要讲究一些工作方法，适当使用一些沟通技巧。

参考答案：

我认为可以从以下几个方面来做：

（1）在测试技能和个人魅力上得到开发人员的认可；

（2）注意保持日常的良好沟通；

（3）在争论时注意表达方式，充分尊重开发人员。

题目 2：一位优秀的软件测试工程师应当具备什么品质？

分析：这道题你可以根据自己的理解来发挥，而只有真正喜欢测试的朋友，答案才能让人信服。

参考答案：

我认为一位优秀的测试工程师应当具备以下品质：

（1）良好的专业技能，包括软件测试和软件开发能力；

（2）积极的态度；

（3）勤奋；

（4）愿意与大家分享；

（5）团队意识；

（6）耐心；

（7）细心。

题目 3：请介绍你在一个有代表性的测试项目中的经验和教训。

分析：这道题的意图是想检验你在以前的工作中的状态。如果你不是一个用心的人，你就很难总结出经验和教训。所以，具体是什么经验和教训并不重要，关键在于你能总结出来。对于没有测试工作经验的朋友，也不必灰心，你可以写实习的项目，或是在学校做过的大作业，甚至到了实在没有办法的时候，你就写从书上看到的项目，但要保证真实，你所要表现的是：你是用心的，你在将来也会用心地去工作。

参考答案：无，请你自己给出答案。

5.7 逻辑推理题

这类笔试题是从网上发现的，觉得很有意思，拿来和大家一起讨论。

毫无疑问，用人单位的初衷是想通过这类题看看候选人的逻辑推理能力，它的重要性并不会排在最前面，不会因此而被“一票否决”，用人单位只会用这个来做为一个参考。

题目 1：帽子的颜色？

有三个人面试，经理有五顶帽子，三顶白色的两顶黄色的，三个人面壁而立，经理给他们每人各戴一顶帽子，看谁能最快猜出自己头上戴的是什么颜色的帽子。三人回过头来你看我我看你，每个人看到的另外两个都戴白色帽子，开始没人猜出来自己帽子的颜色，突然有人说自己戴的是白帽子，经理录用了他，并问他是怎么知道自己戴的是白帽子的？

分析和答案：为了讨论方便，我们假设这3个人是甲、乙、丙，是甲猜出来自己戴了白帽子。我们来做推理：

（1）甲和乙、丙一样，都看见另外两个人戴的是白帽子。因为一共有3顶白帽子和2顶黄帽子，对于甲来说，去掉乙、丙的2顶白帽子，还剩下1顶白帽子和2顶黄帽子，甲无法猜出自己帽子的颜色。

（2）甲、乙、丙都陷入了沉默，这个沉默很关键。甲看乙、丙都没有猜出来，就说明他们也遇到了不能确定的问题。甲想，如果自己带的是黄帽子，对于乙来说，因为甲和丙是一白一黄，乙猜不出来自己帽子的颜色是对的；对于丙来说，如果甲是黄帽子，而且乙猜不出来，那么，丙应当能猜出来自己的帽子是白色，因为如果是黄色的话，乙就会知道自己帽子的颜色只能是白色。所以，如果甲带的帽子是黄色的，肯定会有人猜出自己帽子的颜色。但在这个沉默的过程中没有人猜出来，就说明甲的帽子不是黄色的。

（3）甲可以猜出自己的帽子是白色的。

题目 2：一刻钟？

一根不匀称的香烧需完要一个小时，用两根这样的香怎么测出一刻钟？

分析和答案：这题很短，而“不均匀”三个字加大了难度。如果香是均匀的，我们可以说“对半折，再对半折”（怎么折得均匀还是个问题☺），拿到一段折好的香，点燃，就可以了。可这是不均匀的，怎么办呢？

我们可以这样试试：因为有两根一样的香，我们把它们并列放平，一根从头部燃起，另一根从尾部燃起，到两根香燃烧的部位在一条垂直线上（相遇）的时候，时间应当是过了半个小时。这个时候把剩余的两段香再次并列放平，并开始计时，等两根香燃烧的部位再次在一条垂直线上的时候，就是过了15分钟，就是要求的一刻钟。

5.8 英语

题目 1：请用英语写一个测试用例。

分析：这道题有两个意图，一个是看看候选人的英语基础，另一个是看看候选人是否具备基本的测试知识。题虽然简单，但是如果没有相应的工作经验或者没有做准备，候选人在面对这道题时可能会有点心虚。在选择测试用例的时候，简单的就可以。计算机英语，关键是意思表达清楚，简单的测试用例有利于候选人的表达。

参考答案：

Case: Verify that user can log in MSN successfully using right user name and password

Priority: 0

Steps:

1. Launch the MSN 7.5.

2. Input the right user name, “tester”.

3. Input the right password, “test_*er”.

4. Click the button Login

Expected:

1. The user logins successfully, the main window of MSN is showed.

题目 2：请用英语写自我介绍。

分析：每位候选人的履历在简历里已经写得很清楚，所以用人单位只是想通过这道题看看候选人的英文水平。毕竟英语不是母语，用人单位也不想出难题，“自我介绍”的难易程度比较适中。在做这道题的时候，用简单的词，简洁地写上几段话，要避免语法错误，保持语句通顺，这样就可以了。

每个人都要根据自己的实际情况准备一段英文的自我介绍，我给的参考答案真的只能做为参考，不然，可能冒出N个“小强”来。另外，要背熟你的“自我介绍”，现在一些单位也会考察英语口语。

参考答案：

Hi, I am Xiao Qiang and glad to introduce myself to you.

I come from Jiang Xi province. It is a beatuful place and I love it. I studied Computer Sicence in Jiang Xi Normal Univercity between 2002 and 2006. I like this major as I can face a lot of charlenge and learn a lot. I like to do software testing.

If you ask me “what do you do in the free time?”, my answer is: reading and watching movies.

I am a nice man I think. I like to help my friends and they also give me a lot.

I have good tech skill on computer filed and know how to do software testing. If you can give me an opportunity, I will try my best to do it.

Thank you!

题目 3：介绍你所做过的一个项目。

分析：这个题目比“自我介绍”要难一点，但毕竟也是考察候选人熟悉的领域——以前做过的项目，所以不必慌张，我们要做的仅仅是把中文

的描述换成英语。对于还没有实际工作经验的读者朋友来说，你可以写自己实习期间的项目，或者写在学校里的一些实践。

用人单位想通过这道题了解：

（1）首要的是了解候选人的英语水平。描述得简洁、通畅，就可以了。当然，如果英语方面比较强，那正是一展身手的时候。

（2）了解候选人的实际工作经验。对于项目，应当交代是做什么项目，所采用的技术，项目组的规模，以及你自己的角色。

参考答案：

I have been a team member of ERP project last year. There are about 20 people working for it, 8 tesers, 10 devolopers and 2 leaders. The main technologies used in this project are: C#, Oracle 9i. I was a tester and I owned the testing of sub-systeam HR management. I think I did a greate job when I worked for this project. And also, I learened a lot from this project, such as communication skill, coding in C#, performance testing and automation. I miss this team so much.

阅读我就不再举例了，那的确需要你平时的积累，光看一两道阅读题没有什么效果。但是，面试前再看看英语，让自己对一些单词和语法不至于陌生，把自己的英语恢复到“历史最好水平”，也是必要的。

06

第6章 有问必答——从他人的问题中得到经验

在《软件报》上做了一年的关于软件测试方面的连载，以及我的《软件测试实战—测试Web MSN》这本书出版后，一些朋友根据我留的邮件地址给我来信向我咨询。我觉得这些信写得都很好，他们遇到的一些问题也有代表性，所以我做了一个整理，把一部分的来信和我的回复列在这里，供大家参考。我们只能经历自己的故事，但却能从他人的故事里得到经验。

6.1 软件测试的学习

信件 1：如何学习软件测试

你好！

首先请原谅我这么没礼貌，因为我不知道该怎样称呼您。

我是一名读软件技术专业的大一学生，应该称您前辈吧！我比较喜欢看《软件报》中你写的那个连载文章，对软件测试也比较感兴趣，就是想请教您一下，如果想在软件测试这方面发展，现在开始应该比较侧重哪方面的学习，或者要学好哪些编程语言，如果是自学这方面的知识有什么书可以看……，希望能得您的赐教！谢谢！

WQ

WQ：

你好！

很高兴收到你的来信，从信里可以看出，你是个好学上进的学生。不要称呼我为前辈，我不老☺，也不高深，只是先工作几年而已。

大学一、二年级，还是打基础的时候，这个阶段不要有什么侧重，而是争取把各门功课都学好。软件技术有很多分支，基础扎实了，它能给你提供向不同方向发展的可能性。你提到你对软件测试比较感兴趣，这很好。这方面的书挺多的，你可以到书店去挑一本，原则上是挑简单的，复杂的或者高深的书不适合入门者，而且简单易懂的书有利于你保持在这方面的兴趣，兴趣是很重要的，就像保存火种一样保存它吧。我的一本书《软件测试实战——测试Web MSN》即将上市，也可以供你做参考。

祝你学习愉快！

信件 2：一名大一学生的疑惑

亲爱的大哥：

您好！

我是一名大一的学软件工程专业的学生，听说学这专业很难，的确呀。我们学院的书记叫我们现在想好将来要发展的方向，说句老实话，我现在还对这个专业不怎么了解，根本不晓得往哪个方向发展，所以很迷茫。我经常去图书馆查看相关的信息，看《软件报》的时候看到了软件测试师的考试，还有别的杂志上也有相关的报道，说这一行业在未来很有前景，好就业，所以我现在考虑将来想朝这方面发展。但是，现在还不太了解它，也不知道要多高的水平，要掌握几门编程语言。总之，现在好像在人生的十字路口，不知道东南西北。而且老师说了，现在不考虑好，到了大三的时候就晚了，就好像是定型了。所以现在向你咨询，希望你在百忙之中抽出时间来帮帮我这个迷惘的少年。希望以后还可以向你咨询，先说声谢谢了。

LB

LB：

你好！

来信收到。可能现在真是变了，就业越来越难，这种紧张感从社会和老师那里传递到了你这位大学一年级学生的身上，说实在的，这有点不公平。大学生活是人生中一段美好的时光啊，怎么一开头就是苦恼呢☺。不过，你有点言重了，大一并不是“人生的十字路口”。在面对问题的时候，首先我们要平静下来，去掉一些外在的影响，来看看问题的本质究竟是什么。

你现在对未来的发展方向感到迷茫，这是很真实的感受，也是再正常不过的，因为你刚上大学不久，刚刚开始接触这个专业，谁能在这样一个人生点上就对自己的发展方向说得条条是道呢？怕是找不到这样的人。大学生活最具魅力的地方在哪里，而让那么多人对它都拥有美好的回忆？从

我个人的角度出发，我想，是因为大学提供了一个好的平台，让我们以最低的成本去接触、去尝试各种新鲜事物，奠定事业的基础，以求知道自己的发展方向是什么。所以，我的建议是，你可以立足于自己的专业，列举出可能的发展方向，例如，软件开发、软件测试、网站开发、质量保证、软件工程管理，等等，然后逐一地去琢磨、去尝试，看看究竟什么是适合自己的。

在学校里如何去尝试，可能会是个问题。因为你们有学业，不可能跑到外边去上班。你可以在课余的时候收集一些描述这些职业的资料，然后到图书馆或者书店去找相关的书籍，自己在宿舍里或者机房里亲身体验一下，什么是软件开发、什么是测试？等等。做过以后，你对自己的发展方向会有一些更深入的认识。

这些事情，只是在课余做就可以了，不要希望说一两个月就搞定，要慢慢去体会。能较早地确定发展方向当然是件好事，但从我的观察来看，在毕业后的几年内找到适合自己的发展方向也为时不晚。另外，你也可以突破专业的范围去思考，并不是说学了软件工程专业就说明它最适合你。

你知道吗，发展方向也是会发生变化的，因为在不同的时候有不同的社会环境，你对发展的问题也会有不同的理解，不可能说你定了一个发展方向，这辈子就不变了。在发展如此快的社会中，这种可能性很小。

最后，我想提醒你，对你来说，在这个问题之外，还有更重要的学业，而在学业之外，还有美好的大学生活。祝你学习和生活愉快！

信件 3：软件测试需要什么认证

你好！

我很想了解关于软件测试员的信息，软件测试需要什么认证吗？那要学些什么？在哪里报名？请告之，谢谢了。

WD

WD：

你好！

如果说是想要从事软件测试工作，认证并不是必须的。在找测试相关工作的时候，公司大多注重的是能力。当然，有一个证书肯定比没有的好。我们国家现在正在推行一个“软件评测师”的认证，如果你感兴趣，可以通过网络搜索到一些相关介绍。另外，还有一些培训机构开设了测试培训课程，你可以留意一下你所在地的媒体广告。很抱歉，我不能向你推荐培训机构，因为我没有亲身体验过，不好信口开河，以免误导你。

如果时间紧张不能去上培训班，或者觉得费用偏高，你可以考虑自学。一本书，一台电脑，你就可以开始了。碰到什么问题，你可以再来信，我们一起讨论。

祝学习愉快！

信件 4：软件测试如何入门

蔡先生：

您好！

我是06年计算机专业的本科毕业生，还没找到工作，正在极度迷茫之中。我想从事软件测试方面的工作，又苦于没有基础。现有以下几个问题想请教您：

（1）软件测试怎么入门？有没有必要参加一些价格不菲的培训？

（2）在软件公司中软件测试员的地位和发展前景怎么样？

（3）软件公司需要的是什么样的软件测试人员？我怎样达到公司录用的标准？

期待着您能在百忙之中给我这个迷途的羔羊指点一下迷津。

谢谢！

未名

未名：

你好！

我知道，现在找工作挺难的，我也曾有过四处奔波找工作的经历，所以我能理解你现在的心情。你咨询的几个关于软件测试方面的问题，我会一一回答你，供你做参考。

首先，关于软件测试怎么入门的问题，我相信答案你也知道——学习。参加培训是一种有效的方式，但要看你的经济承受能力和时间安排，如果不合适，还有其他的方式，例如自学。软件测试入门并不难，我相信作为计算机专业本科生的你，只要你愿意学习，是一定能够胜任的。你可以到书店去买几本软件测试的书看看，了解软件测试是做什么的，怎么做软件测试，看看自己是否适合做软件测试。而且，要在面试之前做这件事情，不打无准备之仗。

其次，关于测试员的地位问题，我不知道你是否有某些担心，其实没有必要。在这个竞争激烈的社会里，大家都是平等的，不会有地位低下的职位，只有不称职的员工或者不合适的工作。软件测试是软件研发中的重要角色之一，不可或缺。至于它的发展前景，我个人认为，软件测试有很好的发展空间，现在在国内还是处于起步阶段。

最后，软件测试人员的录用标准是什么？这是给我出了一个难题☺，因为不同的公司有不同的标准，老板们不会藏着一份一模一样的标准答案。我尝试着说几点各公司通用的：

（1）良好的计算机基础知识。你的计算机知识应当宽泛，软件、硬件、操作系统、应用软件、软件工程、编程语言，等等，都要有一些基础。

（2）工作细心。软件测试是要尽可能地找出软件错误，如果粗心大意，网眼就太大了，是抓不到鱼的。

（3）耐心。软件测试的一个特点就是需要反复地做相同的验证以确保软件质量，如果你想从事这个行业，你要有思想准备啊。

（4）外语。现在很多测试项目是承接欧美和日本的外包，在英语和日语上都有要求，要求从业人员基本的听说读写无障碍。

祝你顺利地找到工作！

信件 5：如果要做软件测试都要学习什么

尊敬的主编，我在《软件报》上看到了一些关于软件测试的信息，很想了解一下它是怎样的。我是一名大专生，现读大一，计算机网络技术专业。由于只是大专生，一年半后就要出来工作了，所以开始担心起自己以后的发展。我不想让父母担心，所有的一切都要靠自己努力，然后看到他们开心的笑容。我看到报纸上写软件测试有很大的缺口，因此有意于朝这方面发展，所以恳请主编花费一点时间回个E-mail，解答一下我的疑问吧。还有，它具体要学些什么？

祝工作顺利，身体健康！

Hnuo

Hnuo：

你好！

在回答你的问题之前，我要告诉你，我不是主编☺，我是软件测试系列连载的作者。

你问我，软件测试是怎样的？我举个例子来说明吧。例如，一家软件公司想开发一个学生信息管理系统，在软件研发的过程中，需要测试工程师来检查需求规格说明书是否正确？检查开发工程师有没有按照需求规格说明书的要求来开发？测试工程师需要站在用户的立场上看问题，找出各种各样的软件错误（bug），以提高软件的质量。

如果你想成为一名软件测试工程师，你需要具备以下知识：

（1）扎实的计算机基础知识，包括数学、操作系统、计算机体系结构、数据结构、编程语言等等；

（2）软件测试知识，如果你们没有这门课，你需要去书店选择一本书自学；

（3）英语；

（4）还有一些素质方面的要求，例如，良好的沟通能力和团队协作精神，耐心，仔细，等等。

不要担心自己只是大专生的问题，更关键的问题是，当你大学毕业时，你学到了多少东西？只要你足够努力，你肯定行。

祝学习愉快！

信件 6：我觉得测试难以下手

你好，我是刚踏入这个行业的菜鸟，看过你的文章和书后启发很大，但对于具体的工作还是没有一个很系统的思想，觉得难以下手。我对于计算机方面的知识也不是很懂，想问一下有什么样的书可以让我做为参考呢？现在我要测试一个网站都感觉无从下手，感觉要测试的东西很多，在专业语言概述方面有时不知怎么去准确地概括一个bug给别人看。而公司就我一个测试员，真不知怎么去学习，郁闷...还请你能给我些指点呀！

Els

Els：

来信收到。

你的问题是一种宏观上的问题，比较难回答。我尝试着给你一些建议，不知道能不能解答你的问题。

如果说对计算机方面的知识不是很懂，你就要好好补课了，因为计算机知识是做软件测试的基础，如果基础不扎实，如同无源之水、无本之木，工作会很吃力的。要想打好基础，不是看一两本书所能解决的，你最好参照计算机专业的课程设置，系统地进行学习，例如计算机原理、操作系统、数据库、编程语言、数学，等等。

对于工作中的问题，靠上面的方法可能太慢了，你可以考虑参加一个培训班，或者多向做测试的朋友请教。

通过上述方法，把自己的学习和参加的培训结合起来，近期的问题和长远的发展都能顾及到。

以上所提建议供你参考！

信件 7：黑盒测试和功能测试是一个意思吗

老师：

您好！

我是一名本科毕业生，我对软件测试很感兴趣。我看了您在《软件报》上的连载，也买了您写的《软件测试实战——测试Web MSN》的书，经过一个星期的自学，我觉得这本书不错，通俗易懂，同时也让我明白软件测试是一份不容易的工作，逆向思考的问题要很多，算是要做到“绞尽脑汁”的境界吧。这本书主要强化测试思维的训练，我现在还想找有关测试工具使用的书籍来自学一下，可惜书店没有这方面的书籍，而网上的测试工具的操作方法太简单，不够具体，很难继续学下去，自己摸索都要花很长的时间，不知老师有没有出版关于测试工具使用的书籍？如果没有，您能不能告诉我到哪里去看测试工具的具体使用的资料？

此外，问一下书上的问题，黑盒测试和功能测试之间有什么不同？还是同一个意思呢？因为网上说黑盒测试也是功能测试，而这本书上没有这么说，但意思都是差不多，是不是它们之间的侧重有些不同呢？

期待您的回音，谢谢！

祝：工作顺利，身体健康！

吴

吴：

你好！

首先，感谢你关注《软件报》，阅读我的书。

我没有出版关于测试工具方面的书，市场上专门介绍测试工具的书也

很少。在测试工具方面，你不必花很多的时间，了解一下就可以。因为如果是一款成熟的工具，工具总是用来提高效率或者促进工作的，所以它不会很难用。你可以把时间放在编程语言的学习上，使自己有能力去编写一些小的测试工具。

关于黑盒测试和功能测试的概念，它们是不同的，因为分类的出发点不一样。从测试过程中是否需要阅读代码来分类，一般就把测试方法分为白盒测试和黑盒测试，有的还会加上一个类别：灰盒测试。从测试的内容来分，把测试分为功能测试和性能测试，一般把压力测试也归入到性能测试当中。在实际的测试工作中，因为大多数的功能测试都是通过黑盒测试的方法来做的，所以，二者的关系比较近☺。

信件 8：几个测试方面不大清晰的概念

你好！

我在一家公司做网管，没什么发展机会，公司又是做建筑的，老板跟本不懂IT业，说我们学IT的都没用，所以我辞掉了工作，自己在家学习，准备找一份软件测试工程师的工作。因为自己学习方法不当，或者说根本不知道测试是怎么回事，所以每次笔试的机会就这样以失败而告终。我一直在看你在《软件报》上的连载，在7月2日我面试失败后去逛海淀图书城的时候发现了你写的这本《软件测试实战——测试Web MSN》，我买了一本。在书里我学到了很多与课堂书本不一样的东西，让我对测试有了一定的了解，我看了两遍了，上次在A银行软件开发中心笔试的时候，就是因为看了你的这本书让我通过了第一次的笔试，我非常高兴，只可惜在面试的时候因为我没有测试经验，结果又失败了。

在书里你提到了build,可是管理build的软件是什么？有没有管理build的软件呢？按照软件开发的重要性（由重到轻）来对软件开发的七个阶段的排序是怎样的呢？手机测试和软件测试的测试用例的写法是一样的吗？你会聘请一位没有经验的测试员吗？

最后我要对你说声谢谢，谢谢你的这本书让我知道了什么才是测试。

Maz

Maz：

你好！

感谢你关注《软件报》，购买和阅读我的书，也很高兴收到你的来信。

对于build来说，难的是在于如何管理产生build的源代码，这是一件复杂的事情，特别对于大型软件的研发来说。一般可以通过配置管理工具来实现，例如微软公司的Source Safe。build产生出来了以后，管理倒不麻烦，

准备好硬盘空间就好了。

你问的另一个问题是“手机测试和软件测试的测试用例的写法是一样的吗？”，如果你说的手机测试是指手机软件测试，那么我的答案是肯定的，手机软件测试是软件测试的一个小分支。另外，测试用例的格式并不重要，可能有的单位要求格式A，有的单位要求格式B，重要的是测试思想，你说呢？如果你说的手机测试是指硬件方面的，那就要另说了。

的确，很多单位不会聘用没有经验的测试员，但是，你想，谁不是从没有经验做起的呢？所以，虽然你没有经验，但肯定还是会有一些公司愿意用你，因为有一些小项目会适合你。

你希望我对软件开发的7个阶段排序，我暂时不能回答你，你说的这7个阶段是什么呢？而且，既然是7个阶段，这说明它们缺一不可，都很重要啊。

6.2 软件测试实践

信件 1：测试工作怎么安排

蔡为东：

你好！

测试文档怎么写？测试工作怎么安排？可以提供一些相关资料吗？我是公司负责技术这一块的，非常希望能把软件测试工作做好，但是没有这方面的经验啊！

T

T：

你好！

首先，祝你春节愉快！

测试方面的资料在网上有很多，你可以去找。我有一个建议，如果你们公司有专职的测试员的话，就把这个任务交给他（她）好了，以免过多占用你的时间，我相信作为技术负责人的你还有很多事情要做。从软件工程的角度来看，现在的测试流程（包括测试文档的编写规范）已经成熟，要做的是去实践这些理论，然后总结出符合自己公司实际状况的规程。

祝你们的测试工作进展顺利！如果遇到什么具体的问题，可以来信交流。

信件 2：测试方面的几个疑问

蔡为东：

你好！

关于软件测试有几点疑问，希望能相互交流：

（1）作为第三方的软件测试来说，会采用自研的或第三方的黑盒测试工具，而如果对于小企业来说，无法购买和自研，也无法提交第三方测试（经费和其他问题），此时的软件测试应如何做？

（2）对于测试人员来说，如果不精通编程，对于当前软件无法读懂的代码如何测试？

（3）对于测试管理过程中出现这种情况：任务下发，员工却无法完成，该如何处理？

（4）对于测试管理过程中出现这种情况：任务下发，出现理解错误（未及时沟通），如同你在文中谈到的“千锤百炼SPEC”部分，该如何处理？

呵呵，一下提出了这么多问题，是因为在我所负责的软件测试项目中已经出现，应如何及时有效地处理和完善？

希望和你有进一步交流！

期待你的回复！

L.T

L.T：

你好！

对问题1的答复：

在回答这个问题之前，我想知道，你们的产品是否是没有工具就无法

测试？如果的确是这样，就只能去购买或者自己开发，没有别的办法了☺。如果在没有测试工具的时候也可以做测试或者说可以做一部分，那我们可以安排手工测试。实际上是这样的，测试工具只是实现测试的一种途径，如果手工能做，那结果是一样的，我们做了测试，发现了问题，这样就可以了。

对问题2的答复：

在理论上，我们总是希望测试工程师很强，希望他（她）能够精通编程，而且擅长找到软件错误。可是现实情况总是会有一些差强人意。实际上，这是管理者在招聘人员的时候所给出的待遇的不同所产生的自然结果，在测试人员待遇不如开发人员的时候，优秀的人才都是先奔着开发职位去的。测试主管在面对这个问题的时候，只能去接受和适应，毕竟测试要进行下去。如果测试人员因自身条件所限不了解代码，我们可以从以下几方面做尝试：

（1）鼓励测试人员去学习编程技术，在工作的压力下学习的周期会比较短；

（2）加强开发小组和测试小组的交流，例如让开发人员讲解软件产品的设计思想和代码结构；

（3）对测试计划和测试用例进行审查，让大家在产品还是一个黑盒子的状态时（因为不了解内部结构）尽可能地去扩展测试思路。

对问题3和4的答复：

我想这两个问题是一类的，就一起回答了。

首先我们来看看，问题出现后怎么解决。出现了任务无法完成或者理解错误的情况，作为领导的你要先救火，指责是没有用的，心平气和地坐下来，就事论事。理解错误的，马上澄清。无法完成的，再叫一个人帮忙，先帮着把事情做完。实在没有人了，只好你挽起袖子来自己干了。

然后，我想也是你关注的，就是我们怎么去预防这些问题的出现。问

题的症结你已经知道了，都是沟通的问题。我想，测试领导者就像一个监测仪，需要知道项目的动态进展。关于加强沟通，不同的公司有不同的实际情况，我给一些建议，供你参考，你可以选择合适自己的：

（1）要测试工程师每周给你写工作报告，这样你才能知道实际进展。

（2）做工作检查。例如，一个月一次，你和每个工程师开一次两个人的会，看看他的实际工作进展，避免工作报告中可能存在的有意无意的不实情况。做的时候不要让员工感到反感，要声明只是工作上的检查，大家一视同仁，只是为了把工作做好，而不是不相信他们。

（3）建立测试工程师和开发工程师的沟通机制。理解错误是自然情况，谁都会出现这种情况，因为认知是一个渐进的过程。如果测试工程师和开发工程师能够就一些项目的事情进行讨论，他们在交流中就会得到正确的答案。如果测试工程师总是一个人琢磨，测试的标准都是自己一言堂，会容易出现偏差。

祝你工作顺利！

信件 3：如何做 GUI 测试

齐月女士：

你好！

不知道该称呼你为姐姐还是阿姨，称呼女士吧。我是一名软件测试专业的学生，要写一篇关于GUI案例测试的论文，你能给我点资料吗？我找了很多网站，但是都不怎么明确。关于哪些工具可用于GUI测试，具体怎么测，利与弊，现在国内外的GUI测试的现状，提供有关于这些内容的资料最好。

在此谢过！

ML. W

（备注：我在《软件报》上做连载时用的是笔名齐月）

ML. W：

你好！来信收到。

首先，我申明一下，我是一位男士☺。

你提到的GUI测试，你能从网络上找到很多资料，摆在你面前和我面前的互联网是一模一样的。而且，毕竟是你在做论文，我不敢越俎代庖。我这里给你几点建议，供你参考：

（1）有一些公司提供基于GUI的自动化测试工具，例如Rational、MI等，从他们的网站上肯定能找到相关的文章和帮助文档。

（2）到测试专业网站上去找资料，论坛也可以顺便看一下。

（3）如果关于GUI的中文资料不多，你可以通过google查到很多的英文资料。在看资料的同时还可以锻炼一下英语，看多了就会习惯的。

（4）不懂的地方要向老师提出来，不要不好意思，老师肯定能给你帮助的。

（5）接触一些做过软件测试的人，他们能给你提出好的参考建议。

祝论文写作顺利！

信件4：需求变化快怎么办

你好，看了一些你在《软件报》上的关于测试的连载文章，到书店购买了你写的《软件测试实战一测试WebMSN》，感觉都挺不错的。你的风格是平易的，很适合测试的初学者。

今天我写信来，是因为我现在遇到了一个问题，很苦恼，不知道怎么做。我们公司是做Web应用方案的，实力一般，所以也只能接一些小项目做。我在公司是测试员，我感觉需求变化太快了，有的时候我都没有反应过来，需求又变了。这让我有点不知所措，始终感觉不踏实，不知道怎么样才能做好测试。

你有什么建议吗？谢谢！

WK

WK：

你好！

很高兴收到你的来信，也欢迎你继续关注《软件报》，关注我的连载。

我首先要对你说的一句话，你可能要失望了，你要去适应这种需求快速变化的状况。软件公司可以分为两类，一类是做商业软件，就像微软做Windows、Office，都是刻成光盘卖给客户的，国内的金山、用友等也是这样，其客户群是一类客户，而不是特定的一个；另一类是做项目的，所开发的软件是针对特定的客户，一般规模都比较小。商业软件要求的投入大，所以很多的国内公司，特别是小型软件公司，都是在做项目软件。因为竞争激烈，只能客户说什么就是什么。所以，你要体会公司的苦衷，需求的频繁变化也是没有办法控制的事情。

在这种情况下，测试的处境会比较艰难，因为需求是给我们测试提供标准答案的，现在标准答案都变化很快，我们怎么去做测试呢？我给一些

建议，供你参考：

（1）公司要指定一个和客户沟通的接口，一个负责人。不要让客户直接给某个程序员下指令，那样的话没有办法控制需求，最后软件做成什么样子大家都不知道。这个负责人应当整理客户的需求变化，统一地通知开发组和测试组。

（2）组织人员对需求变化进行评审，测试人员要参加。虽然有的需求因为客户的压力我们不得不做，但公司应当对这个变化带来的影响和风险心里有数。如果有的时候发现风险太大了，应当对客户说明，毕竟客户不知道他的需求变化对程序的影响，不要造成了严重后果以后再说。测试人员更要从测试方面提出风险评估。

（3）测试人员要保持对测试计划和测试用例的更新。需求变化了，测试计划和测试用例都要更新，避免遗漏。

我相信在实际工作中你还会发现一些其他有用的好方法。

祝工作顺利！

信件 5：如何组建一个测试团队

Hi，我是测试行业的新手，目前正筹划组织公司测试部门。在公司没有第二个人了解测试技术的情况下，自己组织和规划测试部门，真是举步为艰。但是，工作还是要进行……，希望能得到行业中前辈们的指点，也可以互相扩展测试思维……

SJ

SJ：

你好！

首先，我要祝贺你走上了测试管理的岗位。我猜你很谦虚，虽然已经被公司委以重任，但仍然说自己是“测试行业的新手”。如果你真是一个彻头彻尾的新手，怕是不会有这样的机会。当然，你也很孤独☺，因为公司没有第二个人了解测试技术。不同的公司，有不同的环境和管理风格，我这里提几点注意事项，仅供你做参考：

（1）虽然你的部门现在还只有你一个光杆司令，但你需要有领导意识。你要意识到，在测试方面，你就是一位领导者，你需要强有力地去制定测试策略，编写测试计划，安排测试进度等等。领导意识是区分领导者和普通员工最重要的因素。从技术转向管理的时候，大家思考问题时容易局限于技术，这是不好的，你现在需要把握测试工作的全局。

（2）阅读一些软件工程方面的书籍，以得到一些对软件测试的宏观上的看法。软件测试的一个重要特点是对软件开发过程和方法有依赖性。如果开发方法不科学，测试工程师工作起来既辛苦又没有什么效果。

（3）争取领导层的支持。软件测试容易给人一种“赔钱的买卖”的感觉，虽然它不是，但只有你得到了领导层的支持，你才能申请到你所需要的人力、物力以及时间。

（4）要意识到做管理工作就是要为部下提供服务。不要当上了领导就对部下颐指气使，这样的领导是不会长久的。软件测试是艰难的，它是软件测试工程师一个用例一个用例地做出来的，测试工程师才是测试工作的真正核心力量。

祝你工作顺利，在新的岗位上做出更好的成绩！

6.3 求职和选择

信件 1：我是应届毕业生，如何去求职呢

您好！首先谢谢您查看我的信。我现在有一些问题不知老师能不能帮上忙或有什么好的意见。我是刚毕业的本科生，学的是电子商务专业，之前有参加过JAVA开发培训，本人在广州，我现在对软件测试的行业很感兴趣，想找软件测试方面的工作，从初级做起，但在求职网站上有不少招聘单位都要求软件测试工作要有1~2年的经验，可我没有这方面的经验，被招聘单位淘汰的机会很大。我现在注重的是找对工作机会，很愿意从低层做起，其他的不重要。请问老师，我在求职方面应该怎么样获得应聘的成功呢？在软件测试方面应该要做好怎样的准备去求职呢？

期待您的回音，谢谢！

Mike

Mike：

你好！

现在的市场行情的确对没有经验的人有些不利，甚至是在简历筛选的时候，如果发现你没有经验，连面试机会都不给，这有点残酷。但是，从另外一个方面来想，谁天生是有经验的呢？每一个有经验的人，都是从你现在这样的情况做起的。所以，首先你要有信心，肯定会有一份属于你的工作。

接着，你要分析一下你的优势。你是电子商务专业的本科生，而且学过JAVA，这很好，你可以尝试着去找使用JAVA的开发项目的测试工作。

最后，你要多看测试方面的书。因为你现在没有直接的工作经验，但可以用书上的间接经验补一补。

以上的建议，仅做为你的参考。

祝你找工作顺利！

信件 2：我应当选择哪一份工作

蔡为东：

你好！

我是一个即将毕业的学生，研究生期间攻读通信专业，现在有一份测试的工作摆在我的面前。因为自己不是计算机或软件工程专业的学生，所以对这个工作不太了解，想和你咨询一下做软件测试工程师的发展前景和路线。有人说做测试很枯燥，横向发展空间很窄，而我又是个没有计算机学科背景的女生，不知道该不该选择这份工作。都说第一份工作对于以后的事业非常重要，而我在面临选择的时候却对这个工作了解甚少，十分迷茫。还有一个让我难以定下决心的原因是非常不想脱离我所学的通信专业，毕竟希望能学有所用。但目前找工作形势非常不理想，这家公司可以让我留京，工资待遇也还可以。这是我的基本情况，非常希望得到过来人的指点。

谢谢！

Y

Y：

你好！

首先，感谢你的信任。

在信的开头，我要告诉你一个准则：我的建议只是一面之词，不可全信，因为任何人的看法都会是片面的。

我是专做软件测试的，但我所给出的回答却不是局限在这个行业。不存在一个适合任何人的行业，选择行业需要根据个人情况来定。现在就业环境的确不大好，很多人都在找工作。这个时候，我们在做一些重要决定的时候，就要把很多的因素综合起来考虑，而不是只考虑一个因素。你想留京，想学有所用，还希望待遇好一些，希望……，这些都是很正常的

想法，只是现实要求我们要把它们排排队，看看究竟什么是最重要的。你说说看，如果你定义一个优先级的话，什么是最重要的？什么次之？什么次次之？还有什么其他的想法？你把这些都列在纸上。人的思路在纸面上会更清晰一些。你有没有朋友？肯定有，好，找几个知心的，到一个方便聊天的地方，把你写给我的信的内容也就是你的想法，都说出来给他们听，听听他们的看法。如果大家有不同的看法甚至争论得面红耳赤，那么你要恭喜你自己，在这个过程中，你可能发现自己能够做决定了。我们在陈述自己的想法和倾听不同的看法的时候，能够更清晰地看到自己，看到自己想要的东西。

实际上，人生不只是第一步重要，而是每一步都很重要。有了你在走这第一步时候的这种谨慎，相信你的未来会更美好，因为你在很认真地生活。

祝顺利！

信件 3：我面临的选择

您好！

我是刚毕业的女研究生，已经签约呼和浩特的一个大学，现在在联系深圳慧通测试。因为联系深圳慧通测试是按社招联系的，他们说现在不能解决户口，一年后解决。我担心他们解决不了，他们说一定能。还有就是工作挺累的，工资4000多，在深圳不高吧。你觉得我该怎么选择？不知道测试发展咋样，还有就是我是女生，一般人认为做老师比较合适，能提供一些建议吗？

谢谢。

迷茫中的人

自称迷茫的朋友：

你好！

来信收到，感谢你信任我。

不要着急，特别是在这样的大事面前。我先问你几个问题：

（1）你的专业是什么？你喜欢你的专业吗？

（2）你喜欢在什么地方（地理位置上）工作？

（3）你希望在什么性质的单位工作？

（4）你喜欢测试吗？你对测试有什么了解？

（5）你自己喜欢做老师吗？

（6）你现在在呼和浩特吗？应当不在深圳吧。

（7）在你考虑选择工作单位的时候，还有没有什么其他因素需要考虑的？例如想离父母近一点，等等。

之所以问这些，只是想知道你究竟想要什么。

祝你顺利！

您好！

感谢您的回信。

我的专业是自动化，本来挺喜欢本专业，想考研后搞点技术，结果我来大连上完研究生之后才知道，这边的课题不是计算机的就是理论的，所以我们有两条路走，做软件和大学老师。我喜欢好一点的城市，但不是北京那样的，压力大，累，大连这么大正好。

我想找控制类的工作，试了几次失败了，课题不合适。我不太喜欢做老师，更讨厌编代码。虽然觉着做销售挺好的，但人家还喜欢找男生，无奈。

我面试深圳慧通测试，面试官对这方面介绍之后，我觉得挺感兴趣，不用编代码，没有研发累。就是找错，而且有经验了还比研发干的时间长。

我现在大连，正在决定是去呼和浩特做老师，还是去深圳做测试。我的在西安华为工作的同学建议我去西安慧通，那里消费低，还没问单位能不能改去西安。

我兄弟姐妹多，父母有人照顾。工作地点由我自己定，没有什么太多需要顾虑的。

谢谢您提供帮助。以您做测试的经验，你觉得走哪条路好呢？

你好！

你的信我看了好几遍，也在房子里踱了好几圈，可是我发现我还是不会做你给我的这道意义重大的选择题：

A. 去呼和浩特做老师　　B. 到深圳或者西安做测试

既然给不出答案A或者B，为了避免交白卷，我只能给你说一点点我自己的感受，供你参考。

（1）消费低不能成为一个影响你的走向的因素。消费低，生活比较舒

适，但经济上的自由度也低。消费高，压力大，工资的起点也会高。

（2）你认为测试比开发容易干得长，其实就你现在可以选择的机会而言，毫无疑问，老师这个职业最稳定，比测试更容易干得长，你能一直干下去。

（3）城市规模的大小不应当成为一个因素。

（4）宣传测试不用编代码是不对的。如果你想成为一个资深的测试工程师，编码和阅读代码是少不了的。

（5）如果你看重户口，那么他们承诺的一年后解决户口这个问题应当可以写到合同里。从法律上来说，口头承诺没有任何意义。

你说你喜欢你的专业，也尝试找了控制类工作，只是没有找到。在梦想没有实现的时候，我不知道有没有可能找到一个离梦想不远的工作。因为我们可以在这个离梦想不远的地方工作一段时间，为实现自己的梦想做准备，也算是在就业环境不好的时候的一种“曲线救国”的策略吧。

哪一个离你的梦想更近呢？

祝你顺利！

信件 4：职业生涯的方向

我的朋友，您好！可以这样称呼您吗？近段时间工作上遇到一点儿困惑，有些两难，不知道可否给我一些行业的建议。希望没有打扰您的时间与工作。在此谢过……

目前我所在的公司有80人的开发团队，目前公司的设计架构是：一个总监，一个总监助理，三个部门经理，五个项目经理。现在公司有六个项目在开展。由于公司成立时间不久，很多的工作都没有正常化，都处在建立阶段。之前许多的工作都是总部进行总体规划与安排。目前，公司在城市A和B分别成立两个点，但同属一家公司，在管理上相对自由得多了。我目前遇到的问题是，公司打算让我在自己的职业生涯里面从两个方向中选择一个：测试与项目管理专员。

我个人认为，如果做测试的话，毕竟和自己的专业背景有所结合，退一步说，毕竟哪个软件公司都需要测试人员。而选择管理专员，可能自己的经验积累以及可控性上不及选择测试。但从薪水及将来的发展上来看，可能管理方面会带来一些可观的收益，以及在个人提升方面有优势。我现在有些为难，老总希望本月底可以给他答复。

做为一个行业先行者，可以给我一些建议让我权衡一下吗？

谢谢！

HY

HY：

你好

首先，感谢你的信任。我没有能力做你人生十字路口的交通警察，因

为只有你自己才有这样的权利。我说几点我的看法，供你参考：

（1）在做选择的时候，合适的就是最好的。你觉得哪一个更适合你呢？毕竟我还不了解你。

（2）我不知道“项目管理专员”的实际职责是什么，如果只是项目经理的一个助手，管理一些文档，不做也罢。不管新的岗位是管理方面或者技术方面，它的难度都应当要高于现在的工作，因为这样自己才能得到锻炼。

（3）在考虑管理职位的时候要小心一点，你可能也有这样的体会，一个管理者只有让人信服，他（她）的管理才会顺利。管理者不是要高高在上、颐指气使，而是要掌控项目的进展和团队的方向。你觉得你现在有这个底气吗？

（4）只要你有实力，从远期来看，做技术和管理的收益是一样的。

祝你顺利！

信件 5：我年龄比较大了，还能学习软件测试吗

你好！看了你写的软件测试的一篇文章，想请教一些问题。我年龄有些大了，对前途比较迷茫。我挺喜欢软件的，但，起步晚了。懂点电脑基础知识，英语基本不懂，编程语言我连是什么都还不懂，看了你写的文章，冷汗直冒，本来还想学软件测试的。能给点意见吗？学这个是不是需要很长时间？有正规的学校培训吗？学费是不是很贵？

谢谢你了……

Liu

Liu：

你好！

很高兴收到你的来信。

你能不能把你的情况详细地说一下？包括专业背景、地区、年龄、现在的职业、对未来职业的各种想法和考虑，等等。这样我可以给你提一些建议。

我今年26了，对于搞电脑的来说，自己觉得已经比较大了。专业？没什么专业，初中毕业之后一直没念了。现在在我一亲戚这儿卖电脑，他们总告诉我要学硬件，可我对硬件没什么兴趣，更喜欢软件一些。我在网上查了一下，好象软件测试需要的专业知识很强，像我这种情况学起来应该很困难吧！还有就是我们这没有这种培训学校，要到外地去找，如果找这种学校应该注意些什么？麻烦你了……

实在不愿意一生就这么下去……

Liu

Liu：

你好！

仔细看过你的来信。

你挺上进的，这是很好的，希望你能保持这种热情。

我觉得你的情况有点特殊，我并不认为你去学习软件测试是一种好的选择，因为的确如你所说的，会比较困难。而且，因为你基础薄弱，学习周期会比较长。

不是每个人都需要上大学，也不是每个人都需要做计算机这个行业。你在选择新的职业目标的时候，你要静下心来，问问自己，什么是自己的优势？现在的社会竞争很激烈，只有从自己的优势出发才有成功的希望。每个人都有自己的优势，例如行业经验、思维方式、待人接物的方式，等等，这个和学历没有关系。

如果你觉得自己思考这个问题很难，就找几个朋友一起商量。

条条大路通罗马，祝你成功！

信件6：我是否适合做软件测试

齐老师：

您好！

真的不知道该如何感谢您，首先非常感谢您在百忙中抽出时间来回复我的上一封信，是关于文科生能否转行去做IT技术的，正是您衷肯的建议使我有了下一步的方向。现在我找到了几个软件测试业内的人聊了聊，还在测试论坛看了一些资料。五一放假时我在 MSN上碰到您，在您的建议下我把《软件测试》（ Ron Patton著，机械工业出版社，网上评论这本书适合初学者阅读）这本书仔细看了一遍，下面我想结合我的自身情况详细向您咨询一下。

先把我的自身情况简单介绍一下吧：

男， 27，本科，专业：经济管理，英语4级，01年毕业，工作5年。

曾从事的工作有：汽车销售、导游、A公司电话客服、税控软件安装工程师、物业等。

可能和IT沾点边的就是税控软件安装了，其实这个工作是我临时的一个经历，前期经过公司的培训掌握税务控制软件的安装操作，然后到客户那里去进行安装调试。这点和软件测试还真有点相关，面对各种不同配置的电脑和打印机，负责把软件安装调试好。

还有在A公司的经历与软件测试也有一点儿相关性，当时公司接了一个项目：电话核对数据库里的数万条公司信息。技术部编写了一个可以外拨电话和存储信息的软件，为了保证使用，我们全体几十个员工一起进行了压力测试。一开始软件的性能很不稳定，手工点击时经常死机，还有其他的一些小错误，经过几轮的测试和改写使软件得以正常使用。

我对自己的性格也在思考，我是 AB血型，网上对这种血型的表述对我几乎都符合，其中提到性格有些矛盾、爱较真儿、对感兴趣的事拥有极大的热情和动力，等等。从性格来讲我希望每天都能很充实，都能进步一

点。我从小动手能力比较强，喜欢存在一些变化性的工作，不能容忍墨守成规的状态，书中也提到IT工作就是一个边干边学的过程。比如前不久我新买了一个诺基亚手机，和我以前的手机操作完全不同，就像刚买的其他硬件设备一样，我饶有兴趣地研究了它几乎所有的功能，遇到不懂的就打电话问客服。

我现在做的物业工作比较偏向于事务性的工作，工作两年半了，我感觉自己没有形成核心竞争力，而且自己在迅速地贬值，随着年龄的增长有种危机感。我和同事没有共同语言。

从开始有对职业方向进行选择的想法后，我做了很多的探索：测评、找专家、自己摸索，等等，测评报告也指出我应该向技术类这个大方向发展，其中也提到了很多的具体行业。

职业推荐：技术工程师、测试工程师、技术培训师、专业领域编辑、硬件工程师、结构工程师、网络管理员、程序员、质量检验员、教师、给排水工程师、康复治疗师、按摩治疗师、生物工程师、药剂师、牙科医生、地质专家、天文研究者、环境专家、体育教练。

但是我从文科背景向理工科行业转换面临的问题比较突出，有人提出反对观点：一是隔行如隔山，对很多专业不了解，难于选择；二是缺乏专业背景，难以迅速入行。很早我就听说过类似的老话儿：由理转文易，由文转理难，IT是年轻人的战场，等等。但我似乎又有些偏执，或者说是别无选择，大学学的知识太过强调理论，造成就业形势不如应用类技术人才。对此我的理解是，现在的 IT行业“英雄不问出处”，讲求的是“即插即用”型人才，如果选准合适的切入点，应该还是有希望的。

我读完书后对软件测试的体会更清楚一些，首先软件测试的书似乎不像传统的 IT技术书那样难懂，它似乎把方法、技巧放在一个很重要的位置，当然软件技术知识是坚强后盾。书里的内容我能看懂 70%，其中比较感兴趣的是配置测试、易用性测试，可以说我对这些的理解还不错。但是像白

盒测试等要求专业计算机背景的章节我理解起来就有些困难了。我的电脑水平一般，应用的主要是办公软件类等等。

我还有些疑问，比如如何选择培训学校？等等，我想这些是后续需要讨论的话题了。我还有一个不太成熟的想法：我想成为搞技术里能说会道的，以及能说会道里面搞技术的，当然这是需要长时间积累的。

我是看了招聘网站上宣传软件测试缺乏人才才知道这个行业的，也是抱着试试看的心态去了解的。由于对IT 行业不了解，凭我自己的探索也非常有限，也许还有其他方向可以考虑。所以借此机会诚心诚意地向您请教一下我的职业定向。知道您平时工作很忙，但是还是想麻烦您能帮我指明一条道路。我现在天天魂不守舍，为了工作的选择实在是发愁，家里也着急，我自己更是着急。在此 套用看到的一句话表达心情："期望，有一天早上醒来，发现阳光明媚，我找到了自己安身立命之所，可以积极快乐地投入工作，那将会是怎样一番景象啊，期待中……"。

到那时我将不胜感激！谢谢您！期待您的指导！

Cai

给您写过信后，我又有些开始茫然了，不知道下一步该如何进行，恳请您的指导。

还有就是非常期待你的那本关于MSN测试实战的书，非常想对照着练习一下，主要是想感受一下真实的测试工作是如何进行的。我现在非常着急，不知道自己的方向该如何把握。有的人的性格和某种工作非常匹配，希望您看了我的信也能帮我把握一下，看我是否适合做软件测试，谢谢！

知道您非常的忙，但是真的希望能得到您的指点！谢谢

Cai

Cai：

你好！

虽然我今天回来比较晚，但我想你应当是在等待着回信，所以还是坐下来给你回信。我理解你现在的心情，你在为自己的职业走向发愁，我也曾经经历过。在这忧虑的背后，我能感觉到，你很真诚，这很好。在黄昏里痛苦地挤压前额的人，将会是第一个迎来美好的黎明的人。

你的来信很长，里面也包含了你很多的疑问，我这里先做一个简要的回答。在你继续往下看之前，我提醒你，不要以为我说的话就是对的，我只能代表一种观点：

（1）我不认为血型和职业有什么关系。

（2）你的专业背景和工作经历不利于你找一份软件测试的工作。但同时，这并不意味着你不适合做软件测试，它们之间没有必然的联系，谁会有天生的测试经验呢？

（3）职业评测也仅仅是代表了一种观点，可以用它来开拓思路，但不必局限于它。

（4）你说你能看懂《软件测试》的70%，这很好。但从你感兴趣的是“配置测试、易用性测试”来看，你还处于一个很浅的层次。在现实的软件测试工作中，更普遍的是做功能测试。

现在的就业形势很严峻，相信这你也知道。虽然你现在做着一份不尽人意的工作，但毕竟衣食无忧。不要轻易地放弃工作，因为没有工作的时候那种内心惶然的感觉是很难受的。我给你一些建议，供你参考：

（1）在考虑新工作的时候，一是看自己喜欢什么，另外就是看是否能找到工作。现在虽然对软件测试工程师的需求比较大，但恕我直言，你的专业背景和工作经验没有强的竞争力。在这个竞争残酷的社会，我们需要发现自己的长处，凭自己的长处来拼得一席之地。你觉得你转行做软件测试是在发挥自己的长处吗？如果不是，你的长处是什么？什么职业更适合

你？

（2）最终确定你是否适合做测试的惟一办法是去做实际的测试工作。但如同我上面说的，你需要保持现在的工作，在这种条件下你如何能够得到一个实践的机会呢？你可以试一试能不能找到兼职的测试工作，没有报酬也可以啊，因为你需要检查自己是否适合做这一行。再多的考虑，也需要到实践中去才知道孰是孰非。为这件事请几天假也是值得的（当然不要影响现在的工作），因为在确定职业走向的时候是需要付出的。

至于你提到我的书《软件测试实战一测试Web MSN》，它将在本月下旬在各大书店有售。谢谢你的关注。

祝你顺利，朋友！

信件 7：我是文科专业的，可以学软件测试吗

我也正在考虑学软件测试呢，可是我是文科专业的，电脑只是一般应用水平，像我这样的能学吗？还得学些基本的软件编程内容，我的理科可不太好哦。还有现在的测试培训学校也不太多，有培训学校A、培训学校B，等等，到底哪个好呢？学费也是差异较大，从5千到1万的都有，我去过培训学校A（1万，周末班，学习期为6个月），但仍是迷茫……。这个行业目前倒是很缺人，不过会不会因为相对较低的门槛，而像软件工程师、网络工程师一样迅速充斥于市场呢？

希望能和你成为朋友，说不定以后还是同行呢！

祝工作顺利！

Jason

Jason：

你好！

职业的选择是严肃的事情，所以我不敢在这里说你适合做软件测试或者不适合，因为我不了解你。你学习的是文科，并不代表你就做不好软件测试，同样，不是说学理科的或者计算机专业的就适合做这个，都是因人而异的。的确，现在培训的学费都很高，也不能轻易尝试。我想总结一下，你现在的疑问是，自己适不适合做测试？如果你认为自己适合，可能培训的花费你都会觉得是可以接受的。

我的建议是这样的，你可以通过几个途径去了解这个行业：

（1）买几本测试相关的书看看，买入门级的就可以。

（2）到网上测试相关的论坛去看看，或者发帖问问。

（3）找到自己身边做软件测试的朋友聊一聊，咨询一下他们。通过朋友介绍的也可以，直接的接触很有用的。

以上的途径都花费不多，但挺有效的。等你知道了自己是否适合做这一行，下一步就好走了。

至于你关于这种职业是否会“迅速贬值”的担心，我觉得没有必要。你觉得现在的软件工程师、网络工程师泛滥了吗？远远没有。只要你有真才实学，工作机会会追着来找你。你现在可以集中精力了解清楚自己是否适合做这一行。

祝你工作顺利！

信件 8：计算机真的不适合女生吗

蔡老师：

您好！

冒昧地打搅您，真的很抱歉，希望没有耽误您宝贵的时间。我是您的读者，看完《软件测试实践——测试Web MSN》很是受到启发，书中的语言幽默，而且讲解深入浅出，容易理解。我原本觉得软件测试无从下手，实在是太抽象，看到了您的书后慢慢有了些理解。但是心中还是有很多疑惑，所以希望能够得到您的指点。

我是一名计算机专业大三的女生，想做软件测试这方面的工作。听说研发方面的工作需要很强的适应能力，而且变化很快，压力很大。但是也听说测试方面的工作比较稳定，需要比较专业的技术，相对研发来说，就没有因为变化太快所带来的那么大的压力，是这样的吗？我的上一届选择工作的学姐几乎没有从事计算机行业的，都选择了其他与计算机无关的行业，她们说从事计算机行业实在是太辛苦，都选择转行，计算机真的不适合女生吗？

殷切希望得到您的答复。

Z-LH

Z-LH：

你好！

你的来信我收到了，谢谢你阅读我的书。从事计算机方面的工作的确辛苦，但也不是那么夸张，不然，怎么还会有这么多人在这个行业里呢，是吧？☺。计算机行业的工作不是体力劳动，性别差异不大，主要还是看各人的能力。在软件测试这一方面，女性很多，而且说不定女性还有优势，因为测试这份工作对耐心和细心要求比较高。我就有不少做测试的女性朋友，而且做得都还不错。你如果喜欢测试，就去好好学，会有机会做软件测试的。相信自己！

附录A　我在微软做软件测试外包

2004年的夏天，我参与的一个大型IT项目结束了，这个项目为期一年，我是管理者之一。我做得有点疲惫，心里也认为比较失败（我的另一部书稿《胶着——我所经历的一个大型系统集成项目》详细记录了这个项目的全过程）。我想，我能不能找到一个公司，去看看究竟什么是成功的软件研发，什么是成功的软件测试。而后我得到一个机会去微软做外包服务，角色是STE（Software Testing Engineer软件测试工程师），虽然只是做vendor，但是是在微软公司内部工作，可以直接参与到微软的软件研发工作中去。毫无疑问，微软能为我的疑问提供答案，至少是一部分。微软之所以能独步软件业天下，靠的是软件产品。所以，它的软件研发过程和软件测试策略肯定有过人之处、独到之处，值得我们学习。

这一年中，我一边做项目，一边又作为一个旁观者在观察。毛子恺大概表达过这么一种说法，他说，作家一边在过着自己的生活，一边又是一个局外人，能够时时刻刻在观察着自己的生活。我不是作家，虽然我很想☺，我把认为对自己或者别人有借鉴意义的东西记录了下来。在我写作的过程当中，我很兴奋，因为这些都是非常好的经验，对于软件测试工程师，对于有志于改进软件工程和公司管理的人们来说，都很有借鉴意义。

实际上，这些答案都不是什么高深的道理，但是，实用，且易于推行。

我相信，你一定会在阅读的过程中有一些感触，这些感触就是改进和提高的星星之火，对吧？倘若本书对你的个人发展或者公司成长有一丝帮助，我会欣慰不已。

希望你们阅读愉快，我的朋友们。

面试

通过北京XX科技公司的面试后几天，我得到通知，要去微软面试。我必须通过XX和微软两家公司的面试，才能得到一个去微软做测试的职位（vendor）。

我记得网上有一些关于微软面试的介绍，描绘得都很神秘，这些无疑在我的心里为微软蒙上了一层朦胧的面纱。我已经工作四年了，内心也积累了一些能量，可以对抗可能的自卑或者自傲，所以并没有特别的激动，因为我知道，我是我，微软是微软。到微软面试也是一种荣耀吗？当然不是。但我心里有一种新奇，因为毕竟这是第一次与微软近距离的接触。我在大学毕业前曾经想到微软实习，却不知道怎么联系。现在才知道微软每年都会接收大量的访问学生（Visiting Student），当然也知道了自己当年的水平不够格☺。

进入知春路上的希格玛大厦，发现微软的确有实力，这幢大厦只有7层，三层、五层和六层都是微软专用办公区，大厦一侧的电梯专供微软使用。XX公司的测试部经理也特意赶来了，她很希望我能够面试成功，因为XX公司在微软亚洲工程院的工作人员还很少。就我自己而言，我也很希望通过面试，到微软工作。虽然只是做外包，但我就可以知道，微软内部究竟是怎样组织起来的？软件项目的处理流程是什么？这个职位是值得期待的。

微软有一个特别之处，非内部工作人员到了前台接待区后无法自行进入办公区。通往办公区的大玻璃门总是关闭着，需要刷卡才能进入，挺有意思。距离能够产生一种美，在这种场景下，距离产生一种神秘和吸引力。当然，更实际的原因是因为这些玻璃门属于软件巨头微软。

这里解释一下vendor这个词。汽车厂的周围都有一大批做配套的厂商，提供零配件，微软是软件巨头，周围也有一大批的公司为其提供服务，这

些公司就是vendor公司。像我这样，来自XX公司，在微软内部为其提供软件外包服务，也简称为vendor。

一般来说，微软对vendor的面试有两轮，分别由不同的技术考官主考，两个人都同意录用方为通过。主要的考察面有两个：一方面是计算机技术，我面试的是STE（软件测试工程师），所以技术方面侧重于测试技术；另一方面是英语，因为微软的工作语言是英语，如果基本的英语听说读写能力都欠缺，日后无法与同事协同工作，当然不在录用范围之内。

英语考核比较简单，方式也没有什么特别之处，无外乎中英互译、口语对话等。技术类考题则是五花八门，不一而足。例如，一个函数String IntToString（int nParameter），请你论述你的测试方法，5分钟的思考时间，然后陈述。作为一个测试工程师，发散思维是很重要的，要考虑很多种情况。这个问题不难，可是，要回答得让考官满意却不容易。这种题没有一个标准答案，考察的是面试者的思维方式和能力。

不要以为技术考题都是这个类型，微软把面试的权利给了普通的员工，每个考官自己出题，这样很好地保证了面试的技术难度，又避免了考题的雷同。所以，有很多的人都说微软的面试很难。这种面试也有一点弊端，那就是势必造成面试的标准不一，出题难度不一。两个水平相当的人，可能一个通过了，另一个却不能，只是因为面试官不同，难度不一。这种误差，似乎任何面试制度或者考试制度都无法避免。我在这里想说的是，不管在什么公司面试，如果没有通过，找自己的原因是应该的，但也不必太过于沮丧，因为总有一部分不是个人能力所能及的，例如面试官的随机搭配，考题的随机性，甚至面试官的心情。我们只要汲取经验，以良好的心态去准备下一次面试就好了。

或许是幸运，我通过了这次面试，得到了到微软工作的机会。

报到

到微软上班之前，我在XX公司上了两个星期的班。这两周我补了补英语，温习了一下C语言，因为听说要做的项目可能需要用到C语言编程。

公司给我介绍了一位英语老师，同事HX，他是计算机和英语双学士，很厉害。我的英语纯粹是中国式的英语，很生硬。我写出来的英文句子有的时候他都看不明白，真让我惭愧。我每天写一篇短文，他来修改。我觉得这种方式很好，让我逐渐恢复了一点对英语的记忆。他现在已经离开了XX公司，我很感谢他，祝福他诸事顺利。

接到微软的上班通知后，公司给我买了台新的台式机，由我送到微软。为了避免盗版软件和病毒可能造成的纠纷和危害，微软需要把vendor的计算机的系统盘格式化，重新安装操作系统。Vendor离开微软的时候，整个硬盘都要格式化，以防泄漏机密。做法是残酷了点，可也是微软维护自己权益的合法之举。哪怕是一个小软件，里面也凝聚了很多人的智慧和汗水，来之不易，所以希望你支持正版。

两天后，我正式到微软上班。简单地办理了一些入职手续，我得到了一张门卡，作用就是用来打开那些大玻璃门。门卡的一面有我的照片、姓名和身份，都是打印上去的。然后我被带到自己的座位上，并被告知，一会儿我的老板会来看我。当时我觉得这个称谓很有意思，老板，现在想来，这也的确很贴切。按照微软的组织结构，一个普通员工，只需要对自己的老板负责，其他的部门都是不相干的，很少打交道。有的时候，另外一种问你的老板是谁的方法是，“你report给谁?”，你给谁写工作报告当然谁就是你的老板了。

我的Leader（也就是我的老板）带我见了见同事，然后我得到几个文档，让我熟悉一下项目。我的vendor生活就此开始了。

研发构架

相信大家对微软的研发构架有所耳闻。在一个软件产品的研发团队中，微软主要有三种角色，他们是：

（1）PM（Program Manager，程序经理）。他们负责提供软件产品的需求规格说明书，与销售人员和客户进行接触，以获得足够的反馈，让软件产品跟上市场需求的发展。产品并不一定要求应用尖端的技术，但是，一定要与客户的需求相吻合。良好的市场反应是他们追求的最高目标，也就是要能卖得出去，能挣钱。微软这个巨头能听取客户的意见吗？当然。我想，能让微软听取你的建议并能在某一款软件中体现出你的需求，是一件令人兴奋的事情。自然的，对微软而言你也必须是足够重要的。我所在的项目就针对GE公司的反馈而对一个模块做了大的修改。

（2）Dev（Developer，开发工程师）。他们负责把PM的需求规格说明书变成实际的软件产品。在整个团队中，他们技术能力最强。他们的意见对PM有很重要的参考价值。例如，如果PM描述了一项功能，Dev认为从技术上无法实现，PM就要认真考虑一番了。当然你放心，开发人员不会滥用这项权利，哪个专业的开发人员会轻易说自己实现不了某一个功能呢？☺。另外，在微软内部有无数个开发团队，他们会互相提供咨询服务，PM不会轻易被某个懒惰的开发人员所蒙骗。

（3）Tester（软件测试人员）。他们负责把PM的需求规格说明书和Dev研发的软件产品相对照，发现错误，监督和验证错误的修正，提高软件的质量。微软在测试上舍得投入，在我们这个项目里，Dev与Tester的比例甚至达到了1:2，这是以前在国内的企业内工作时想都不敢想的。

三个小组各自有一个Leader，负责自己小组的管理，三个小组之间没有从属关系。Leader也做具体的技术工作，例如Dev Leader也要编写模块，PM Leader也要写需求规格说明书，测试组Leader也做测试。三个小组并驾齐驱，

这样在根本上保证了各自的独立性，特别是保证了测试的独立性。我很欣赏这种制度，它在根本上保障了测试人员和测试小组的权利。同时，由PM做日常工作的推动者，推动项目向前发展。

这即是所谓微软的三驾马车。

知识共享

在微软，当遇到一个技术问题时，你没有机会说这个问题没法解决，因为你知道，哪怕再难，在微软内部也肯定有人知道答案。但是，微软这么庞大，人员分布世界各地，你怎么得到答案也是个问题。同时，我想，微软的管理者应当会经常思考一个问题，就是怎么尽可能快地实现知识共享？

如果你想跟踪某一技术的发展，或者在学习一个新的工具，你可以参加公司内部的Group（讨论组）。例如，关于自动化测试，你可以参加Test Automation Group（自动化测试讨论组），这样你就能收到大家讨论测试自动化问题的大量邮件。很有可能，别人在讨论的问题也正是你想了解的。你也可以向任何组提出你的问题，只要你能够找到正确的Mail地址。热心人是很多的，一般你都能即时收到满意的答复。

面对一个问题，技术人员的第一反应可能就是自己埋头钻研，特别喜欢攻克问题的那一刻的喜悦。而在微软，你受到的鼓励是先向别人发问。先把问题描述出来，发给相应的Group或者同事，然后你再自己去想，这样相当于有多个线程同时工作，效果更好。所以，怎样清晰准确地描述问题，是个学问。

在公司内部创建几个讨论组的成本是多少？成本为0。只要系统管理员在公司的Exchange Server（微软的一款知名的邮件服务器软件）上添加几条记录即可。让几个热心且技术强的带动一下，及时回复大家的邮件，只要

能够给大家一种“有用”的感觉，这些组就有了生命力，延续下去不难。有的人可能会说，我公司小，没有几条“枪”，没有这种必要的。其实，有了讨论组，发个通知也方便啊，不必要在发邮件的时候一个一个地去添加邮件地址了。公司越小，人力成本就越可观，让大家迅速实现知识共享，减少对新雇员的需求，不就是降低了人力成本吗？

在微软，如果你需要一个测试工具（微软大多只使用自己员工编写的工具）来辅助测试，首先想到的应当是是否已经有现成的软件，而不是自己马上捋起袖子写一个。微软内部有一个专门提供测试工具的网站，一搜就知道了。微软内部有不计其数的内部网站，只要你想学习，你能得到巨大的知识财富。一位获得亚洲工程院最佳测试奖的同仁说，微软是知识的海洋，此言不虚。

另外，内部刊物，无论是电子的还是纸质的印刷品，也是实现内部知识共享的一个好途径。

知识共享是一个软活，不容易出成绩，但是，时间长一点，不论公司大小，就能显出功效来。

失误

在项目组工作一段时间后，我们每个人都得到了具体的任务，成为the owner of features，也就是具体功能的拥有者，负责人。老板强调一个owner（拥有者）意识，就是说，如果你是一个功能的owner，你必须对这个功能负责，我相信他传达的是微软的一种文化。这样除了能给人一种荣誉感，更多的是责任到人。冤有头，债有主，谁有问题谁负责。日后关于软件的质量从客户处得到反馈后，每个错误都可以进行追查和反思。我一直相信，不管是公司还是社会，都需要责任落实到人，集体无责任是很可怕的。

我的工作任务之一是做UI测试，这个我从老板的邮件中已经知道了。

在我们项目组里，对UI测试有自己的定义——做非功能性的交互界面测试，例如界面布局是否符合要求，各控件、各字符串是否符合要求，是否符合用户使用习惯，最大化/最小化后界面是否正确，在不同分辨率和不同界面属性设置下的显示是否正确，等等。但是，同时我却收到了另一个功能A的需求规格说明书，我就有点疑问，我究竟是做UI测试还是要做这个功能的测试？我想问个清楚，但老板已经到桂林开会去了，要一个星期后才回来。我需要在这个星期里把测试计划写出来。我想了想，决定写功能A的测试计划。没想到，这却犯了一个错误，而且是一个比较大的错误。老板回来后，发现我做错了，马上指了出来，要我改正。在后来的谈话中，我逐渐意识到，我是错在没有及时沟通。有疑问要及时发问，只要问过了，如果他下错了命令，那是他的过错。如果你有疑问而不及时沟通，错误全在你。这算是我在微软工作的第一个教训。同时，及时与老板沟通，让你对老板透明化，任何时候老板都知道你在做什么，这也很重要。

我在做自己负责的两个feature的测试的同时，我接受了一个任务，做自动化测试的尝试。微软内部有一个测试工具，可以用来做录制和回放，适合做客户端方面的UI自动化测试。但是，这个工具用在我们这个项目却是有些勉为其难。因为录制回放工具一般适合于客户端与服务器端交互的程序，而我们产品的基本工作场景却是两个客户端进行交流。我编程的功底也是比较普通的，测试程序做出来后，在两次公开展示中都出了问题，不能顺利运行。在规模越大的公司，同事之间的关系会越正式，一个人仿佛就是一个单位，公开展示是彼此了解和协作的一个重要方式。对任何一个展示都要好好准备，对于它的影响力你一定要有清醒的认识。

后来我这个任务被中止了，虽然没有的批评，但得到的一个委婉的评价是："编程能力不是很强"。我当然知道这句话的含义，心里有点特别不是滋味。尽力去做，却没有做好。后来测试组采用的方案是写了一个win32程序充当假的客户端，像个机器人，能够对特定的消息做反应，这样做自

动化测试时只需要考虑一个客户端，问题就大大简化了。我自己觉得，这个失误是因为自己技术的短缺导致解决方案有天生缺陷，方案有问题你怎么做也是白费力气。

我是平静地在陈述自己的失误，甚至略微有点高兴。因为我认识到了自己的不足，我就会在这一刻获得进步。或者您读过一遍以后，也略有触动，那是再好不过了。

千锤百炼Spec

不知道大家是否有埋怨软件文档的经历？软件需求和开发文档不完善或者根本就没有，始终是软件开发过程中的一大障碍。文档的不完善对测试来说更是一大伤害，因为测试需要标准，这些标准需要写在正式的软件需求规格说明书(Spec)里，不能只存在于一些人的脑袋里。

在微软，PM（Program Manager）对软件需求负责。PM负责做市场调查，与客户交流，把对产品的要求详细地记录下来，回答软件工程中规定的“需要做什么？”的问题。正确和详细是对软件需求规格说明书的两大要求，而形式不拘，文字为主，图片为辅。我们要承认PM也是普通的人，他们也会犯错误，而如果需求规格说明书中存在着大量的错误，势必导致整个软件项目的失败。所以，需要在制度上去减少PM犯错误的机会，这种制度在微软就是Spec Review。Spec Review就是召开一个需求规格说明书的审查会，开发和测试人员都要参加。每个与会人员在会前仔细阅读需要审查的需求规格说明书，提出自己的一些改进意见，并在会上讨论。改进意见可以是纠正其中的错误，也可以是表明自己的观点，说某些地方不具有技术可行性或不完善，甚至只是对某一点表示怀疑，自己也不能肯定对错。什么都可以说，这是促进改进工作的的好方法。如果要求大家只能提出正确的改进建议，怕是时间会浪费在各自斟酌自己的看法是否正确上了。大

家畅所欲言后，由当事人来选择性地接受其中的一部分。会后，PM把采纳的修改意见整理出来，发Mail通知大家，对Spec将会做哪些修改。这个过程提高了Spec的质量，而且让大家都参与了进来，让整个项目的成员都理解了需求，大家都在一个进度上，实现了“同步”。所以，在软件开发过程中，这种方法很可取。

在接下来的开发和测试工作中，还会继续发现Spec中的某些问题、错误或者无法在规定时间内实现的功能或指标，我们都会积极地去和PM沟通，让他们去决定是否需要对Spec做出修改。这种方式体现了对Spec的尊重。在一个软件项目中，需要保证Spec的指挥地位。是的，Spec也会有错误，但请执行它。发现了错误，请提出来，PM迅速去改正。Spec是标准，项目需要标准。

如果你是一个开发人员，你却说：“PM对技术了解不多，这个功能设计不合理，这个地方我不会按照Spec去做。”，这种做法是错误的。如果你没有让PM知道你的看法，你就是默认了SPC，默认了就要去执行它。

如果你是一个测试人员，你却说：“这个功能在Spec里面定义不详细，我按照自己的理解去测好了。”，这也是错误的。如果你发现Spec某些方面定义不清晰、不完善，请提出来，让PM去更新Spec，这是他们的工作。如果按照你自己的理解去做测试，风险特别大。你怎么知道你的理解一定会得到PM和Dev的支持？而且，说不定，你自己在不同的时间会有不同的理解。这样，你如何保证测试的严格性？测试不严格，有可能会成功吗？

在各方的努力下，Spec经过千锤百炼，始成真经。这个时候，产品也出来了，与Spec一一对应，煞是喜人。

测试实验室

在M2（Milestone 2第二个里程碑或第二阶段）前期，同事B到美国工作

一个月，回来后给我们带来一组图片，拍的是微软美国总部一个测试实验室。从照片来看，这个实验室规模大，布线整齐，显得很专业。他在邮件里说，他非常惊讶和羡慕他们的实验室，希望将来在北京也有这么一个高水平的实验室。我们在北京也有实验室，虽然和照片里的美国总部的实验室相差很大，但有一点是共通的——微软的确舍得在测试上花钱。

以我们这个项目组为例来说吧。大家都知道微软的MSN，现在都成了办公一族的常见联系方式了。而微软还有一个产品，类似MSN，提供给大企业，让这些大企业有自己的实时消息系统，保证了系统的稳定性（因为在局域网内的产品要比广域网里的稳定得多），而且也保证了私密性和安全性（不让内部信息通过公网传输），这个产品叫Live Communications Server，简称LCS。LCS有两种客户端，一种是Win32程序，叫做Communicator，是Windows Messenger的升级版；另一种是基于Web的，叫做Communicator Web Access。我所在的测试小组就是负责测试第二种客户端程序。我们北京测试组一共有10个人，分工很细。我们的工作用机除外，微软还购买的50台左右的品牌机来做测试，加上一些旧的计算机，有60台的规模。另有2台HP的服务器，2台Load Balance，3台Mac（苹果机，国内多见于平面设计行业）。特别是这个Load Balance，听说1台要20万人民币。这种花费，的确需要有钱的公司才能承受，也的确是看重测试的公司才愿意承受。

就我个人的体会，有一部分公司，虽然设立了测试部门，在测试条件上却“手紧”，使得测试部门成了公司淘汰设备的中心，有的公司即使是淘汰的设备来做测试也不够用。“巧妇难为无米之炊”，没有软硬件条件，测试怎么能保证质量？

这里可能有一个问题，明眼人马上就会说：“我们没钱。微软多有钱啊，我们怎么能跟微软比？”。的确，我们不能和微软比，而且简单地进行数字比较没有意义，因为项目各有不同。但是，如果你是一位公司领导，

你是否给了开发和测试同等的财权？自己的事情自己知道，不要在嘴上说测试重要，请从行动中体现出来。

测试环境的搭建

测试环境的搭建在测试工作中容易被忽视，有些时候就成了“隐含工作”，通常是算做是测试的准备活动，各自去做。放松了对测试环境的管理，风险是很大的。稍大一些的软件产品都是依赖于一系列软硬件产品的，例如OS，.Net Framework，Browser，AD或者其他的，测试时，必须保证这些底层的测试环境是正确的。而且，软件测试是严谨的，宣称支持的每个平台都需要经过测试，没有经过测试的就不能说自己的产品支持它。

我们小组的项目是一个中小项目，300到350个人月左右（包括了测试量），测试环境的搭建工作是比较繁重的。因为我们要支持多种OS，多种Browser，多种AD构架，这几个条件组合起来的集合是可怕的。要提高测试覆盖率，就要及时更新测试环境。除去30台客户端，我们有20台左右的Server，开始新一轮测试的首要条件就是根据规划把这些Server更新好。从环境搭建的技术上来说，安装各种不同的操作系统很简单，但我们还需要搭建AD、Exchange Server、Load Balance、Firewall Server、LC Server，如果没有专人来做的话容易出现混乱。从M2起，老板在测试小组内部成立了一个Deployment Team，即环境搭建小组，4个人。虽然都是兼任，但是环境搭建被计算为工作项目，不再是“隐含工作”。Deployment Team的工作任务就是及时更新Server，Client的更新则还是交给大家去做，因为Client的更新只需要更新操作系统，没有难度，不容易出现偏差。

我对测试环境的搭建有以下几点建议，供您参考：

（1）把测试环境的搭建计算入工作内容，由专人全职或者兼任来做，责任到人。如果一个人在几天内做了环境的搭建，不要说他什么测试也没

有做，搭建环境也是测试工作。如果环境没有准备好，或者不符合要求，你怎么去测试？

（2）对于环境搭建的具体要求，一定要有文档记录下来，这是PM的事情。同时，也要有文档记录实际环境是什么样子（应当由测试人员来完成），因为肯定有一些扩展的规定，例如IP的分配，Server name，用户帐号，等等，有文档则便于查询。

（3）对于环境搭建中遇到的问题以及解决方法，都要及时记录下来。古人说，一文钱难倒英雄，现在计算机的知识面这么广，也会出现某些细节难到高手的时候。有的时候是一个命令，有的时候是一组特殊操作，费了九牛二虎之力找到了答案，但时间一长就会忘记，很可惜。好记性不如一个烂笔头，记录在案，方便以后查阅参考。这次花了时间，就节约了下次的时间，也算是珍惜了自己的劳动成果。

导师

微软很注意从内部挖掘潜力，导师制度就是其一。如果你是一个正式员工，进了微软后公司会给你指定一名导师。导师和你之间的关系是松散的，只是他应当比别人回答你更多的问题。一般地，两个人一个星期吃一次饭，AA，随便聊一点什么。交往多了，导师就是你的一个朋友，能给你很多帮助。这样，微软不用花钱，就能让新员工迅速融入新环境，实现更迅捷的知识和文化的承接，实在是妙。

这种不花钱的好制度，为什么不学习呢？

我进入到微软工作后，因为只是一个vendor性质的STE（软件测试工程师），所以没有享受到导师制。但是没有关系，事在人为，后来我主动申请参加另外一个同事和他导师的见面活动，并从中获益。他的导师是个“国际友人”，从美国到中国来暂时工作，我们谈技术，谈生活，得到了很好

的英语口语锻炼机会。

1:1

在第一次接触到1:1 meeting这个名词的时候，我心里知道是什么意思，却不知道怎么去读出来，还好边上有人拉了一把，说：“这是1 on 1”。

1:1 meeting 是指两个人的会议，面对面地交流，一般都是指老板和员工之间的固定而又非正式的会议。我们项目的测试组有10个人，其中1个是老板，我们每个人和老板的1:1是两周一次，一次半个小时。在1:1中没有预定的议题，老板大概问问这段时间的工作情况，员工可以畅所欲言，谈谈自己的感受，提出自己的建议。会议的形式和气氛都很好，没有什么压力。这样，员工除了正式的会议之外还有一个渠道与上级沟通，管理者通过这个1:1“体察民情”，增强了公司内部的相互沟通。在公共的会议上，大家都只会就事论事，很少掺杂个人情感在里面，而每个员工都是独特的，管理者要想深入地了解员工的实际想法和精神状况，就需要另外一种途径，1:1是一个很好的选择。

1:1没有什么指定议题，如果你特别忙，可以主动取消，老板也不会怪你的，还有下次嘛。

同事离职

同事J在工作了一个多月后，主动提出了离职。

我们俩座位在一起，经常一起去吃午饭，相对来说走得近一些。对于他的离职，我感到很惊讶，但也理解他。他不大习惯于vendor的身份，感觉与微软的正式员工不一样，所以当有另外一个机会的时候，他选择了离开。对于vendor缺乏归属感的问题，我自己有比较深入的体会。虽然我们在微软工作，但不是微软的员工，我们很少回自己的公司，对自己的公司也没有

什么认同感，这样，两边都搭不上，好像一个河心的小岛。

Vendor和正式员工，工作关系紧密，正式员工当然也不会对vendor居高临下、颐指气使，毕竟是同事，天天见面，不会弄得不好看。而且，真是处得熟悉了，做技术的人之间没有那么多事，关系还是挺好的。但是，毫无疑问，不说隔阂，区别还是存在的。我就碰到过几件事情，让我感觉到不同类别之间的员工的区别。对这些小事情，我的态度是，不能深究。如果你将这些事情上升到人格受到侮辱的高度，连身边的朋友也认为是这样，但，接下来又能怎么样？要别人道歉？要公司给个说法？没有必要。记得一位去美国进修的官员在一本书中写道，他去一些公司申请参观的时候受到了冷漠的对待，甚至遭到歧视。可是，他想，他是来美国学习的，不能在“别人的态度”上纠缠，继续去做后面的事情好了。我到微软是为了学习，它是软件巨头，在我面前是一个海洋，有那么多好的经验我可以去分享，我很感激它给我这个机会。欲成就大事者，不可拘泥于细微小节。而且，哪个公司或者团队内部不会有一些有意无意的误会或者摩擦呢？公正地来说，发生的这些事，还是属于正常范围的。

另外，从客观的角度看，因为vendor没有归属感，所以他们更容易敏感，内心也脆弱一些。有些事情的发生可能本来没有什么特别的意思，但是因为每个人的立场、观点的不同而使理解各有差异。就好像测试工程师抱着找错的念头去测试就容易发现问题一样，如果你真的是时时处处想着“是不是被区别对待”，就容易有不好的体验。

我理解同事J的离职，并祝福他找到更好更合适的工作。我则会继续做下去，到学成为止。

我个人认为，无论在哪个大公司，Vendor和正式员工的最大区别是vendor没有培训机会。公司没有这个义务对vendor进行培训，毕竟vendor不是他们的正式员工。每次我看微软的正式员工去参加培训我都羡慕得流口水，每次又都只能自己咽下去。要改变这种状况的主动权完全在自己的手

里，只要自己足够强，微软的培训大门也是向vendor敞开的，同时，也会有其他优秀的软件公司提供的培训机会。

好好努力，多学习。机会会有的。

入职培训

在微软工作半年后，得到一个培训机会。虽然这次培训姗姗来迟，但我还是挺高兴的，因为我很看重培训。而且，我对微软的培训体系很感兴趣，想看看究竟。所以，虽然那次测试课程是面对初次介入测试的新员工，我还是兴致勃勃地参与了。

培训共两天，两位老师各讲一天。从培训的内容上来看，没有什么特别，都是一些测试相关的理论和实践。它的独特之处在于：

（1）微软为测试做了充分的准备。编写了教材，编写了课堂用的被测程序，搭建了测试环境，使得我们每个人都有动手做的机会。

（2）课堂形式比较好，以实际操作为主。计算机本来就是一门操作性很强的学科，不劳老师那么费劲地把理论讲得天花乱坠。

（3）老师比较强。我记得有一位老师对Windows有比较深入的了解，介绍起来侃侃而谈。

我想，对于每个公司来说，我们也可以让我们的培训更有实际意义，是吧？

Performance review（工作效率检查）

微软会在固定的时间点对员工做Performance review，好像对vendor是每季度一次，对正式员工是半年一次，我不是很确定。这里的Performance不是指某个产品的性能，而是人工作的效率。这种方式并不新鲜，国内叫做评审，很多公司每年做一次，叫年终考核。

在微软内部做Performance review大概的流程是这样的，首先是员工填写一张表，写出对自己这一季度工作效果的评价，并要分别陈述优点和不足之处。然后老板会填写上他（她）的评价，最后做一次面谈，面谈是老板把他（她）的评价对员工公开的一种方式。

我觉得三个月的时间点选择得比较好。任何一个项目，三个月过去了，都会有变化。相应地，一个人在三个月当中肯定做过了一些可以列举出来的事情，否则就是碌碌无为。三个月正式地总结一次，能及时发现问题，及时纠正。如在年终考核中发现了问题，错误已经铸成，来不及纠正了。考核、评审的目的是什么？是希望项目、公司能够做得更好，而只有及时发现并改正错误，错误所带来的损失才会降下来，项目的成本才会降下来，成功的可能性才会更大。年终考核还有一点不方便的地方，因为事关重大，牵涉到对一个员工一年工作的评价，操作起来略嫌沉重。如果把考核分解到每个季度，大家都能心平气和地坐下来谈，氛围轻松一些。年底时把4个小考核累计起来就可以了，大家也会心服。

据说微软正式员工如果两次Performance review都不能通过，得到的评价比较糟糕的话，就要离职，主动的或者被动的。可见允许穿拖鞋上班只是文化的一面，管理还是很严格的。

CC（Code Complete，编码完成）

现在介绍软件工程的书籍很多，如果你细心的话，相信你能从软件工程的理论中看到一个基本的方法：软件开发要分阶段来做。而现实工作中，常见的也是令人恼火的一种情况是，好像开发人员无时无刻不在挥汗如雨地编码，直到项目结束。这样，软件开发就陷入了编码的泥潭，编码成了主导，其他都是附庸。不知道什么时候项目结束，不知道项目所处的阶段和状态，编码占据了所有的时间。

在这一点上，微软能告诉我们什么？

在微软，开发小组在每一个里程碑中有一件重要的事情要做，就是声明Code Complete。在编制开发计划的时候，CC Date是一个很重要的日期，会着重标出。进入开发阶段后，所有人的眼睛都盯着CC Date。例如说，一个项目的M1（Milestone 1）为3个月，那么CC Date可能定在第二个月的30号，所有的M1的功能都要在这个日期前完成。经过测试小组的CC测试后，开发小组可以公开声明，他们的M1已经CC，然后就进入测试阶段。

CC的确定，有效地保护了测试。如果不能通过CC测试，开发小组申请延期，压力全在开发小组。CC以后，测试人员一旦发现没有完成的功能点，一律按照bug处理。如果开发人员不努力的话，你就看着bug数噌噌往上长吧。

CC虽然是由开发小组声明，但是必须得到测试小组的首肯，只有通过了CC测试以后，才能公开声明CC。如果开发小组随意地声明CC，这和没有CC这个检查点没有什么区别。所以，对于这一点管理者要把好关，开发小组声明已经CC必须得到测试小组的许可。

微软是很看重CC的，我们的项目在M1声明CC的时候，微软亚洲工程院院长还特地写信来祝贺。

Dev对bug的批评

第一个里程碑（M1）结束之后，在美国工作的4位同事来到中国，做工作访问。其中有一件重要的事情，就是大家一起开会，总结在M1中的工作。会议的气氛很热烈，开发人员对测试小组发现的bug有以下批评和建议：

（1）描述不清。开发人员看过bug报告以后，不知所云，或者有一些不确定的语句，需要让人揣测。这个批评是很好的。测试人员在描述问题的时候，无意中会忽略某些条件，而缺少了这些条件，可能很难重现这个

bug，这给开发人员的修正带来很大的不便。这一点测试人员必须要心服口服地去接受并提高。因为bug的描述是给开发人员看的，如果开发人员不满意，就相当于顾客不满意，不要争辩，去改正就好了。

（2）没有日志（trace）。开发人员认为，有些bug，需要提供程序的日志，否则很难去发现问题的原因所在。在复杂的程序中，尤其是Server端程序，即使测试人员准确详细地描述了错误发生的步骤，可是开发人员还是没有办法知道问题的所在，因为导致出现问题的可能性太多了。这个时候程序日志就能起到很好的作用，因为日志中记录了程序的每一步操作。如果测试人员在描述bug之前，能够对这些日志进行简单分析，提出一些参考意见，当然更好。如果开发人员在准备修正一个bug的时候，还要重新复现一遍，以得到trace，的确浪费时间。

（3）有一些bug是重复的。开发人员指出，有一些bug说的是同一个问题，只是报告人不同而已，这样会浪费开发人员的时间。从测试的角度来看，如果想要减少重复的bug报告，测试人员之间要加强交流，知道大家都报告了什么bug，这样在自己填写报告的时候，至少大概有个印象，觉得某些bug可能已经报告过了。后来我自己发现一个方法比较有效，就是每天把自己报告的Bug都通过邮件发给大家，这样大家在看Mail的时候就会顺便看一下。这样做，减少了我的bug和别人的bug重复的情况。

批评是个好东西，它使得我们测试小组在M2更专业了。当然，只有你欢迎批评，你才能得到批评。

年度最佳测试奖

激励员工当然重要，这一点毫无疑问。可是，用什么去激励？钱吗？当然需要，但不要把钱当作惟一手段。用钱来激励员工有几个不便之处：一个是易比较性，员工可能会拿来与其他同事所得到的做比较，与其他公

司做比较，说不定拿了钱心里还忿忿不平；另一个是钱有刚性上涨的特性，奖金额只能越来越高，而不能降，因为大家的心里预期会越来越高，而这对公司是一个潜在的威胁，有一种被强制的感觉。有没有别的办法，能够避免这些缺陷而又能有效地激励员工呢？大家是否记得，我们上学的时候，尤其是在小学和中学，很多奖励都是离不开奖状的。我妈现在已经退休了，她曾经感叹说，为了那几张奖状，努力干了几十年。这里面的奥妙在于，荣誉激励。微软在这方面做得很好。

微软亚洲工程院每半年做一次最佳测试奖评选。我去参加过一次这样的表彰会，那次评选出两位优秀员工。表彰会很简单，但很热烈。主办方特地请到了微软亚洲工程院院长来颁奖，合影留念。奖杯很精致（现在的工艺水平越来越好，基本上是你要什么样的就能做出什么样的来），更重要的当然是荣誉。在人群中成为耀眼的明星，谁不喜欢？

获奖者还有发言的机会。其中一位的发言使我印象深刻，她说到了两点：

（1）微软就像是海洋，只要你想学习，你能得到无数的资料。

（2）做测试人员，一定要做好与PM和Dev的沟通。

一个奖杯多少钱？比大家出去吃一次饭的费用少多了，但是那次表彰会后，有多少人为之心潮澎湃，希望有朝一日能够站到领奖台上的，怕是不在少数吧。

激励员工的方法有很多，如我们可以给员工发带有公司标志的T恤、衬衫，项目到了一个阶段，可以给员工发印有项目标志的小礼品，花钱肯定不多，只要安排合理，很得人心的。我到微软工作一年，虽然是一个vendor的身份，但已经得到了4个小礼品：一件运动外套，一个旅行小工具盒（梳子、镜子、小镊子之类的），一个耳机和一个小魔方。我很喜欢它们。

测试人员全体大会（All Hands Meeting）

现在想来，参加All Hands Meeting还是一件很有意思的事情。

All Hands Meeting就是全体大会，没有什么特别的议题，大家以项目组为序，轮流发言，互相认识一下。当时ATC（亚洲工程院）已经有八十多个测试员，济济一堂。在会议开始的时候听到主持者说大家都要介绍一下自己，下面一片嘘声，大家觉得似乎不大可能。但是，后来还是做完了，大家都介绍了自己，而且妙语迭出，笑声一浪高过一浪。

有一个人说没什么爱好，只是喜欢喝啤酒。另一个人接着说喜欢和喜欢喝酒的人一起喝酒。结果第三个人站起来说，就喜欢看两个喜欢喝酒的人喝酒，看他们喝醉。下面大笑，感叹这人真是狡猾。

有很多人说喜欢玩电子游戏，喜欢爬山。结果，到了快要结束的时候，一位女生站起来说，我没有什么特别的爱好，但是有两件事情特别不喜欢，一个是打电子游戏，一个是爬山。大有把前面那些说喜欢二者的人通杀之势，痛快，大笑。

还有一位哥们说，没有什么爱好，就是喜欢坐城铁，并邀请志同道合者一同前往乘坐，真是搞笑。

最绝的是一位漂亮MM的发言，她说，她高兴的事情就是能够在这里和自己喜欢的人做自己喜欢的事情。她的本意非常好，就是客气地表示能和大家一起共事很高兴，但是因为这句话容易引起歧义，结果大家报以热烈的笑声，特别羡慕她所在的小组的成员，她把会议气氛推向了高潮。

搏击游戏和观看《星球大战前传3》

不同的文化有不同的鼓励倾向。我参加了几次集体活动，略有感慨。

一次是ATC（微软亚洲工程院）全体测试人员去望京玩搏击游戏。一

组5人，两组对抗。每人有一把电子枪，身上套着电子背心，上有靶心，击中敌方靶心即得分。对抗赛的核心任务是，大家要团结一致，在一个黑暗且烟雾弥漫的环境中，保护自己的大本营，攻击对方的大本营。这种游戏让大家在兴奋中谋求合作，鼓励对抗和竞争。

这种游戏有刺激性，参赛前大家纷纷讨论策略（当然，可能在对抗过程中忘个精光☺），赛后翘首期盼公布成绩，排名靠前也是件值得得意的事情。更有甚者，觉得不过瘾的，在大家都轮过一圈之后，再次披上装备玩第二次。

另外一次是ATC统一观看《星球大战前传3》。那个通知的Mail中有一句话引起了我的注意，它说，“组织观看《星球大战》这类的科幻片是微软的一贯传统。”。影片的确制作精美，那些动画效果简直不知道是怎么做成的，能够激发我们的想象力。自那以后，我对一些能够扩展自己想象力的片子比以前注意多了。

我不敢妄言企业文化的事情，因为这个话题太大了，我懂得不多。我只是感觉这种鼓励合作、竞争和想象的氛围很好，对公司和个人都是件好事情。组织集体活动，除了请员工吃饭外，我们还有很多选择。

Jason

Jason，美国人，30岁左右，高个，光头，英语文学学士，是我的同事，也是我的老师。他和我一样，也是一个vendor（在美国）。因为他的妻子在微软（中国）工作，所以他在微软（中国）得到一个三个月的临时工作的机会。他在这三个月当中，是在工作日时到微软上半天班。这种工作机会或许是微软解决“两地分居”的一种方法。我们所在的项目因为是跨国合作项目，对英语的要求比较高，所以我们就争取到了这一宝贵资源，就是Jason的工作任务之一是检查我们写的工作文档，同时参加我们的会议，为

如何提高我们的口语提出一些建议。

我知道机会难得，现在出去随随便便上一个口语培训班，哪个不要一千大元以上？而且还不一定有外教。我赶紧给Jason发信，除了表示欢迎之外，希望他每周能够安排两次与我的面谈，每次半小时，我希望借这个机会练好口语。因为这个项目我们每周都要和美国同事开会，对口语的需求比较紧急，也比较高。Jason回信说，他的工作任务也很多，时间不够，只能改为每次15分钟，每周两次。我答应下来，这样已经很不错了，在我之后申请的，怕是连我这样的待遇也没有。

上课的形式很简单，我们两个聊一些能够聊得起来的话题。他照顾我，总是说得比较慢，让我体会了一些能够听懂英语的喜悦。他把我不懂的单词用水彩笔写在小茶几上，课后我抄下来，自己去找答案。

短短三个月，一晃就过去了。Jason的工作签证已经到期，他的任务也完成了，他去了上海旅游。很难具体地说他对我有多少帮助，但我知道，很有帮助。语言的学习就是在日积月累中去掌握，你很难说哪一部分更重要，哪一部分不重要。只要积累多了，功到自然成。

我所要强调的是，要主动地创造条件去学习。像这次，如果你主动向Jason申请，你就能得到他的帮助，而且他的帮助会成为他的工作的一部分。有的同事可能就没有主动去联系，可惜地失去了一次很好的学习机会。你可以停下来想一想，自己的周围有没有什么触手可及的资源而自己没有想到呢？

总结会（Post-mortem）

前事不忘，后事之师。微软很懂得这句中国古话的道理。在一个产品研发流程当中，在每一个里程碑结束之后都有一个总结会，Post-mortem。

M1结束后，美国的同事到中国做访问，PM、Dev和Tester全体人员参

加了总结会。M2结束后，北京这一块的开发和测试人员又坐到一起开了一次总结会。

总结会的气氛很好，鼓励大家开口，不做好好先生，认真分析自己和团队的优势与不足，但对事不对人。会议的气氛是微妙的，如果大家都只说好的，互相表扬甚至吹捧，会让总结会流于形式。如果大家互相攻击，把累积的埋怨拿到会上来发泄，会影响团队士气，甚至使得团队土崩瓦解。

我很喜欢这种总结会。大家都是普通的人，但是如果能够深刻地认识到自己的优势和不足之处，并发扬长处、改进缺点，哪个团队不能做出成绩？

据说微软内部在中国开的总结会大体趋于温和，只有委婉的批评，没有争得面红耳赤的，而在美国要激烈得多。有人说这是因为文化的不同，我深以为是。

有的时候我就在想，为什么一些企业不能很好地做总结？为什么一个半年或一年的项目却没有一个正式的有效果的总结会？是文化吗？是，但不是中国文化，是企业文化，很多中国的企业在这一块做得很好。

另外，扩展开来想，对于我们自己，如果真心想提高的话，也是要常备反省之心的。

让人头痛的会议

我的一个朋友曾经给我发过一个Mail，她说：“现在我的工作也有一些困惑，项目启动了，相应的文档，设计，计划很多，在这个公司，每一个环节都需要进行评审，所以现在基本上天天都在开会，每次会议都要3个小时左右，唉~~~。可能是公司小的缘故吧，每次开会基本上参与项目的人员都要参加，会议上大家都发表自己的意见，这样评审很难进行，而且评审不是一次两次就能通过的。项目刚启动就遇到这么多的问题，而项目上

期又很紧张，三个月完成，只有两个月编写代码和测试的时间。现有三个项目同时启动，我心里很着急，可是他们的评审还需要继续进行。 呵呵，向你唠叨了一些，微软在这方面的管理是怎么样呢？在开始研发写代码前，也需要很多的评审会议吧，给我讲讲啊：）。”。

看到这里，我会心一笑，我们也曾经这么痛苦过。

毫无疑问，很多会是必须的，特别是在公司，没有没事就开会的必要。为什么开会让员工觉得不堪重负？我想，最主要的是因为他（她）觉得这个会对自己没有用。如果会议对每个人都很有用，会稍微多开一点怕是也没有人埋怨的。可问题是，怎么能让每个人都觉得会议是有用的呢？缩小与会人员的范围是个不错的办法。

只要求需要发言的人参加会议。如果一个人在一个会议当中，没有必要发言，他（她）就没有必要参加会议。我们的项目在M1的时候每周有一次全体会议（电话会议），北京和美国的同事（包括PM，Dev和Tester）都参加，后来发现这样的会议效果很差。因为项目大，每个人只做一小块，20个人一起开会，很多时候所讨论的内容都有很多人不了解。在M2的时候，管理层做了改进，把会议拆成了两个小会，一个Server组的，一个Client组的，这样会议就有效多了。

同时，管理层鼓励开两个人的小会。两两开会，及时沟通，这样的会随时随地可以开，简单方便。

所以，会议的问题实际上是一个内部沟通机制的问题。因为需要沟通，所以才有了会议。我们需要寻找一个更加高效的开会方式。例如评审会，如果会前没有准备，会议开完了，大家对需要评审的文档都还没看完，这样的评审会不会有效果。我们项目组对评审会的定位是，仅仅讨论存在分歧的地方。如果你有不同的看法，请在会前向会议主持人提出。主持人认为有必要的，就在会上提出来供大家讨论。会后主持人需要给大家群发一个会议的讨论结果，没有结果的会议是没有意义的。

让每个人专注于自己的工作，同时多开一些高效的短会，这是理想状态☺。

Usability Test（用户使用性测试）

记得上大学的时候，有一个同学得到一个让大家特别羡慕的机会，到微软做测试。工作很简单，但报酬不错，而且，随时随地都有可乐可以喝……。现在知道，他是在做Usability Test。

我们这个项目的产品是面向普通办公人员的，所以Usability Test显得很重要。虽然微软有能力很强的PM，他们设计产品的水平很高。但是，微软也明白，不能让一帮做技术的决定一切，产品使用方不方便，还是普通用户说了算。虽然我们在M1实际上只是做了一个产品框架，实现了核心功能，但就在M1后期，公司及时安排了Usability Test。因为用户的反馈进来得越早，越好改进。

我没有参与和组织过Usability Test，据说要全程录像，这样才便于分析用户的反应。虽然说作为测试人员的一个基本要求就是站在用户的立场看问题，但是，测试人员毕竟是技术人员，思维方式和操作习惯还是会与普通的用户有区别。通过看Usability Test报告，我们能够了解产品在UI设计和功能设置方面还有哪些不足之处。

看Usability Test的测试报告很有意思，能看到自己作为一个测试人员忽视的问题。例如，有的用户说，某个下拉列表框中的一个值被选中后，其左侧要有一个小勾，下次再次打开这个下拉列表框的时候就知道上次选择的是什么；还有的用户说，颜色搭配不明显，通知用户的方式不容易被注意到，等等。

我喜欢看Usability Test 测试报告，简单有趣又容易发现自己测试的盲区。

一般来说，Usability Test会对产品的设计有着比较大的影响力，所有合理的用户感觉不方便的地方都会推动Spec的改进，进而产品得到改进。

File Spec bug（提交软件需求规格说明书的bug）

毫无疑问，开发人员在一个项目中的作用是很重要的。我见过两种不好的倾向，一种是开发人员成了项目的主宰，一个功能能不能做由他说了算，项目的进度也控制在他们手里，他们往前走两步，项目就往前走两步，他们往后退三步，项目就往后退三步；另一种是开发人员特可怜，PM和测试人员都来指责他，等待他们的只有恶梦式的加班。这里，我们可以借鉴微软的一个做法，就是测试人员不要忘记给Spec（需求规格说明书）找bug，使得PM、Dev和Tester互相制衡。

比较常见的办法是，测试人员如果对某一点需求有疑问，大家就会在Mail中讨论，直到测试人员明白了，或者PM对Spec进行了修改。那么，如果去提交关于Spec的bug有什么好处呢？

（1）如果提交了Spec bug会有利于开发工作。PM也是普通的人，所以Spec中有前后矛盾的地方或者错误也属情理之中。但是，在Spec的错误被修正之前，测试人员就不必追着让开发人员实现这个设计错误的功能，不要拿着Spec当圣旨。已知的Spec bug和少数正在讨论中的功能，开发人员可以放一放，先实现没有异议的功能。

（2）这些Spec bug的修改也是需要计算成本的，PM本来说做成A，后来发现这是个错误，改做成B，这时开发和测试都有一个讨论时间和响应延迟，在做项目核算或者责任追查的时候，要把这个提出来。有了Spec bug，大家一起总结的时候，就有据可依了。

（3）从PM的方面来说，Spec bug对他们也会有压力，促使他们在编写Spec的时候更加小心谨慎，提高Spec的质量。

所以，测试人员在工作当中，对Spec的基本态度应当是，尊重Spec，以Spec为依据来检查程序，但同时也要擦亮眼睛，去发现Spec中也就是需求定义中的问题，这样才体现出了测试人员可贵的能动性。发现Spec的bug，对降低项目的成本大有好处。

在我们小组的项目中，在M1阶段，测试人员都属于熟悉和学习的过程，对于发现的Spec中的问题，一是不大肯定，二是也没有意识到Spec bug的重要性，大多在Mail中讨论一下就完了。到了M2，测试人员日渐成熟，同时开发人员对Spec的抱怨也给测试组做了提醒，测试人员本着对项目负责的态度，发现并提交了一些Spec bug，远在美国的PM对这些bug很在意，一般都能迅速给予答复，这就达到了高效沟通、降低成本的目的了。

不要忘记更新测试计划和用例

项目的推进总是越来越紧张，直到项目顺利结束或者流产。相对来说，测试工作对技术的要求少一些，但测试工作量是巨大的。因为我们的产品是基于浏览器的，PM列了一个操作系统和浏览器的二维矩阵，在每一个交叉点上，我们都需要执行一部分或者所有的测试用例。在最紧张的时候，有一种疲于奔命的感觉。你没有办法停下来，脑子里有一个声音在说，往前，往前。

需要提醒的是测试管理人员。是的，测试人员必须往前赶，但，也别忘记了看看，测试计划和测试用例是什么时候写的？这一段时间内Spec做过哪些更新？另外，通过测试实践我们又发现了测试计划和测试用例中的哪些不足吗？如果有（哈哈，肯定有的），请在繁忙之余，偷一个“浮生半日闲”，给测试组安排一个时间，让大家停下手中的测试，更新测试计划和测试用例。

测试用例不会是一成不变的，你的一个Case在运行过了5遍以后，该发

现的问题都发现了，如果不补充新的Case，怎么去对付变化中的被测产品？产品每天都在出新的build，你的测试用例却一成不变，风险很大的。

从我们项目的实践来看，在Beta阶段发现的bug中，大概有五分之一是因为缺乏测试用例，在前期（M1，M2）没有被发现而在Beta偶然遇到的。Bug是越早发现越好，这个道理大家都懂，所以，我们要及时更新自己的武器——测试计划和测试用例，增强战斗力。

Update bug quickly（迅速更新bug）

经过我们测试人员的努力工作后，发现了一个bug，在把这个bug提交上去后，要记住，这个时候并没有万事大吉。Bug并不是一成不变的，我们需要及时地去更新它，实际上就是记录对一个产品中存在的问题的讨论和解决办法，我们需要及时回过头来检查，看看这个bug处于什么状态。有的时候，开发人员可能有不同意见，需要我们测试人员提供更多的信息来证明；或者开发人员认为问题已经解决了，但测试人员发现并不是这样，这些情况都需要提供及时的反馈。一个bug的生存周期（从打开到关闭）越短，成本就越小。

微软的bug管理系统是很完善的。一个bug被提交后，PM、Dev和Tester都可以编辑这个bug，陈述自己的观点。大家都可以看到所有的编辑记录，方便交流。

我们这个项目组做的是一个基于Web的提供给企业内部使用的Messenger（聊天工具），在M2的时候，我发现了一个 bug，就是在经过了长时间的闲置后，Messenger的自己的状态会变成“未知”的。因为这个bug需要提供服务器端和客户端的日志信息，所以当时在提交这个bug的时候也是花费了不少精力。后来这个bug给推迟到Beta阶段来修正，这也是合情合理的，因为产品总是一步一步成熟的。在Beta阶段的某一天，这个bug突然

显示为“已经修正”。我翻看了相关的记录，开发人员并没有做专门的修正，只是说：“这是在很老的代码上测试出来的问题，现在应当已经被修正了。请测试人员验证。”，一验证，果然，没有了这个问题。因为这个bug的特殊性，验证需要一个晚上的时间，所以验证一次通过后，我就关闭了这个bug。

可就在第二天的新build上，我，还有其他几位同事都发现上面提到的那个bug又回来了。我当时应当立即把这个bug重新激活，但是因为忙于做Test Pass（测试人员越到后来越忙⊗），而这个问题毕竟是公共区域的问题，我没有看出严重性来，也就没有看重这件事情。几天过去，我还是没有去更新我的bug。

这时，Beta已经临近结束，测试组将要选择一个稳定的build做最后一次的测试，然后公司就要公开声明发布这个产品的Beta 1版。这时所有重要的bug都要及时上报，否则就会有严重的后果。我的一位同事YD本着严谨的态度，收集了上面提到的那个bug的所有信息，及时做了提交。这个bug在Triage会议上得到了重视，要求开发人员马上修正这个问题，否则不会做Beta Release。因为这个贡献，YD得到了两张电影票作为奖励。电影票只是一种形式，重要的是YD的工作得到了认可和赞赏。

那两张电影票本来是属于我的，哈哈，我一直在跟踪这个bug，却在最后关头变得迟钝，差点掩盖了这个问题。如果Beta build带着这个问题出去，后果不堪设想。

所以，请在工作里划出一小块的时间去检查你的bug，及时地更新它们。

一个有趣的bug：压住Enter不放

我们的产品有个查询功能，用户输入一个字符串后，能够在服务器上查到匹配的企业内部员工的名字，这对于大企业来说很方便，用户能够迅

速找到对方并通过发送消息进行沟通。触发查询功能有两种途径：一种是单击Find这个按钮，另一种是直接按回车键。查询功能挺复杂的，因为需要和Active Directory 进行交互，另有一位同事专门负责，我也就没怎么关注过这个功能。在测试的过程中，听到的关于这方面的bug都是功能性的，这也不足为奇，因为功能复杂嘛。

后来，已经到了Beta期间了，在一次会议上听到一个关于这个查询功能的bug，让人大跌眼镜。这个bug是这样，如果用户输入一个字符串后，按住Enter不放，他就能手工地在很短的时间内向服务器发送数量巨大的查询指令，虽然没有造成什么危害，但这是一个很危险的口子。

从思维方式来说，这个bug的产生需要非常规的操作，所以，测试人员的发散思维很重要，套用一句广告语，“不走寻常路”。毫无疑问，这个bug修正起来很简单，怕就怕在开发和测试人员都忘记了。开发人员忘记了还可以反问一句，“测试人员怎么没有发现？”，测试人员忘记了就没有什么借口好说的了☺。

忘记了产品的基本逻辑

这是一件尴尬的事情。我们全力以赴地去做产品，而在特定的场合下我竟然会忘记了产品的一些基本逻辑，让人无可奈何。

在Beta期间，一次我在做Smoke Test（冒烟测试），在我验证了开发人员对bug的修正都正确后，随意做了一些探险测试（Ad-hoc），突然，一个发现触动了我。我发现，在我从自己的联系人列表中删除一个组后，这个组中的联系人并没有真正被删除，而是被移到了一个公共组。如果这个bug属实的话，将是一个特大的bug，因为删除是基本功能之一。我感到很兴奋，或许测试员有重大发现时都有这种感觉。我马上叫来开发人员做确认，他看了看，懒懒地说：这是需求要求的。

他的话像一盆冷水，让我清醒过来。的确，这是需求规格说明书要求的，只是我当时没有想起来。我觉得有点脸红，为了抓出bug，自己犯了一个初级错误。还好没有直接作为一个bug提交，否则笑话就大了。微软的bug管理系统有一道防线，不管你有什么理由，你都无法删除一个bug。在随后的一次会议上，一个开发人员还拿这件事情来说事，言下之意就是不知道你们测试组在做什么，这些基本的逻辑都能忘记。

我不能说什么，不能争辩。单凭这件事情并不能说明我或者测试组不行，只是希望以后冷静一些，尽量避免这种初级的错误，减少开发和测试组之间的摩擦。

一个简单的文字bug

在M2的最后几天，工作量慢慢少了，项目就像是一辆高速行驶的大巴，现在我们准备进站休息几天了，然后精神抖擞地进入下一个里程碑——Beta。就在这时，我收到项目组内一位美国同事J的Mail，他是服务器端的开发人员，来自美国，到中国来工作，中文学得还不错。他在信里说，他发现了一个文字bug，通过这封Mail来通知大家。

在前面说过，我们的产品是基于Web的Messenger，在与服务器连接中断时，我们需要给用户显示一些错误信息。其中有一条错误信息是这样的："Your session has lost, please sign in again"，翻译过来就是，你的这次会话的信息已经丢失，请你再次登录。你有没有发现这句英语的毛病？

J在信里说，这句话有一个简单的语法错误，如果要表达"会话信息丢失"的含义，可以使用"Your session has been lost"或者"Your session was lost"，也就是要用被动时态。他说，英语国家的人看到"Your session has lost"是要发笑的，会显得我们的产品很不专业。是啊，如果一个外国人说"我的钱包被丢了"，我们也会发笑的，不是吗？

要避免这些简单却让人难以忍受的错误，除了测试人员要更细心外，配置一位文本审查员是一个好的方法。所有的文本，都要经过这位文本审查员的审核后才能使用，他（她）约束PM，可以保证PM在Spec中的文本是正确的。这种角色在微软被称为UA。毫无疑问，如果你的产品是以英语国家为基本市场，UA的母语需要是英语。如果市场在中国，UA的母语需要是汉语。

这个bug给我的另一个感受就是，测试需要更小心，更仔细。

沟通

软件业的个人英雄时代已经过去，取而代之的都是团队作战，我相信沟通的重要性不用我再多说什么，真正的问题是，如何做到充分而有效的沟通？

我所在的团队是一个国际化的团队，团队中的成员分别在中国和美国办公，物理距离、文化差异和人员之间的陌生更显出沟通的重要。在团队级别上，我们有两个有效的沟通渠道，一个是小组会议，一个是评审会。通过电话会议形式的小组会议，让小组成员了解项目的进展，认识当前的问题，同时拉近了彼此之间的距离。评审会则是进行不同思想的碰撞的时候，修改后的方案就是智慧的融合。

微软公司在沟通上是花了很大力气。为了避免跨国合作的团队内部的生疏，微软内部有一种制度，每半年左右会安排一次访问。这种访问可以是美国团队访问中国，也可以是中国团队访问美国，“百闻不如一见”，见过面以后，不管是发邮件或者是打电话，都会亲切得多。

个人层面上的沟通是我们自己需要努力的。因为大家的工作都很紧张，不会有太多的空余时间，所以工作安排中不会有专门的沟通时间，每次沟通都是在工作中进行的，具有很强的目的性。例如，测试工程师有了一个

疑问以后，会就这个疑问去找开发人员，两个人讨论一会儿，疑问有了答案，沟通也就结束了。在实际的工作中，我觉得有几种形式还比较有效，说出来和大家分享。

（1）在发现bug后，如果我们测试工程师并不是很肯定，可以先去找开发人员确认，这样至少能避免自己错误理解某些东西。开发人员对这种沟通还是很欢迎的，并不会认为是耽误时间，因为这样他们在一些有疑问的bug提交之前有了一个发言的机会。提交假的bug会让测试工程师面临不小的压力，这种提交前的沟通有利于减少提交假bug的概率。

（2）在测试执行阶段，可以浏览其他同事发现的bug，这样做既能让我了解项目的进展，这些bug也会在我的脑子里留下一个印象，降低了我提交重复的bug的概率。超过一定比例的重复的bug会给测试组带来负面的评价。为了让大家能够方便地浏览我发现的bug，我下班前会把自己今天发现的bug都拷贝一份发到大家的邮件里，这样大家在查看邮件的时候就能顺便看到我发现的bug。这种看别人bug报告和让别人看自己的bug报告的做法，并不是“面对面”的形式，但这是一种有效的沟通形式。同样的，在有空的时候，多看同事的测试计划和测试用例，也能增进彼此的了解。

（3）团队活动是一种很有效的沟通形式，要多参加。有的时候有一种奇怪的现象，可能在建议举办团队活动的时候很多人积极谏言，而真正定下来时，各种不能参加活动的借口就都来了，如“太早了，起不来”，“周末想休息一下”，“陪朋友逛街”，等等。其实团队活动可以使大家不是为了工作而聚在一起，这是个难得的沟通机会。所以，如果有了团队活动，即使有点困难，也要力争参加。

在沟通这方面，我的体会是，真的需要我们在内心中去看重它，并有意识地为沟通创造机会，这样会使得我们的工作越来越顺利。

Performance testing（性能测试）

从理论上来说，测试需要覆盖所有的方面。一般来说，功能测试比较好做，因为它直观，可以直接从UI上得到验证，输入一个值，做某些操作，得到一个输出，正确与否一目了然。而性能方面的测试，需要测试工具的辅助才能完成，比较而言难一些，但不可因为它难而不做它，要迎难而上（不好意思，有点喊口号的味道了☺）。

现在很多的软件产品，特别是行业应用软件，都是Client/Server 结构的，我们可以通过做performance testing来模拟Server端在实际场景中可能受到的压力，观察和验证产品的表现，从而知道产品是否达到预期的要求。Performance也会是开发人员的一个盲区，因为他们也没有条件去做这样的试验，所以需要测试去发现问题，并并给开发人员提出问题，开发人员才会有动力去优化代码，提高产品的performance。如果产品的性能的bug没有被及时找出来，等用户部署后，在上班时间几千人同时使用时才发现性能问题，如产品表现出响应缓慢、拒绝登录、甚至Server端崩溃，这都是让人痛心的事情。

怎么去做性能测试呢？编写测试工具是必要的，但我想，最重要的莫过于定义性能测试的模型（model）。所谓模型，就是你需要测试哪种情况下的性能，例如用户数、用户类型、用户操作类型、用户登录的时间和频率、期望的结果，等等。定义了模型后再用测试工具去模拟出这个模型。如果没有模型，性能测试就可能没有抓住测试的重点。模型应当由PM给出，可以有多个模型，每个模型各有侧重，这样覆盖率就要好一些。

如果项目不是很大，可以让一个人在做性能测试的同时做做压力测试。性能测试是测试符合实际情况的条件下产品的处理能力，而压力测试则专注于测试极端条件下产品的处理能力。例如，如果一个产品通常是提供给1000人以内的公司使用的，你就可以测试一下10000人同时在线时对Server

的压力和Server的处理能力，这就是压力测试。

从我们小组的性能测试来看，性能测试的好处主要有两个：一是让开发人员认识到产品的性能还远远不够，需要去优化代码；二是发现了在用户较多的情况下资源分配上偶然发生的死锁，要知道，一旦死锁发生，产品就不能工作了，后果很严重。所以，性能测试非常重要。

最后提醒一下，不要忘记去衡量Client端的性能。如果你的Server端的性能很棒，而展现在用户面前的Client却慢得像个牛车一样☺，也是不可以的，到时站在你面前的会是一批满腹牢骚的客户。

胶着

有时项目会进入到一种状态，时间越来越紧张，系统却出现了不稳定的倾向，甚至连续几天的BVT（Build Verification Test）都过不了，开发和测试人员的心态也日渐浮躁，项目就好像进入了一个泥潭，我称这种状态为胶着。这时谁都觉得有问题，却不容易说出究竟问题在哪里？什么是解决之道？仿佛全部都粘在一块了，分不开。

在接近Beta结束的时候（项目组的Beta结束，然后才能得到一个Beta版本，公开发布后普通用户就可以试用Beta产品，所以我们平常说的Beta是用户Beta，实际上这个时候项目组的Beta阶段已经结束），我没有想到项目进入了胶着期。

Beta的最后两个星期，Dead line（最后期限）越来越近，手头的任务还是很多。从开发方面来说，还有一些功能要实现，例如支持在一台计算机上同时登陆多个用户，实现新的UI，替换不合适的字符串，做全面的错误处理，等等。同时，还有一些虽然不是新功能，但也需要时间去做的，例如采用正式的文件名，对源代码做加密处理，等等。而在制定时间表的时候并没有为这些活儿单独安排时间，以为只是举手之劳。从测试方面来说，

需要一个稳定的build去做Test Pass。我们的测试量特别大，各种操作系统和浏览器的组合都要考虑到。而且，每一次切换系统，从组合A换到组合B时，服务器和客户端都要刷新配置，动作是比较大的。

测试组对一个稳定的build的渴求，以及开发组连续几次因为一些小问题导致BVT失败，使得项目组内的气氛紧张起来。测试组的时间表本来就很紧张，加上没有得到稳定的build（根据微软的规矩没有通过BVT测试项目就不能向下进行），时间上就更是捉襟见肘了。虽然没有发生争执得面红耳赤的情况，但大家都感觉到了压力。从另外一个方面来看，这种紧张也是好事，至少说明大家都很努力，只有大家都在意才会有压力啊，如果谁都不在乎，大家可能还会因为没有build而落得个自己轻松自在呢。

经过了这段难熬的日子，我想可以从这几方面来对这个情况进行改进：

（1）如果一件事情需要多个人来配合才能完成，那就一定要在管理上下功夫。统一时间，统一标准，一气呵成。例如说前面提到的文件改名，你的动态库改名了，其他的调用者也一定要改。看似简单，就好像骑双人自行车的两个人一样，要步调一致。但一旦有一个开发人员没有跟上，整个的build就不能正常工作，十分可惜。

（2）开发工作是智力密集型的工作，有的时候不好量化，但一定要尝试着去量化。制定出一个预期的时间表，及时核对，这样我们至少知道还有什么没有做到，还需要多少额外的时间。不要让开发人员有可乘之机，什么都在最后一刻完成。这也是在帮助开发人员提高效率。测试人员也是这样，不能什么都留到最后来测。

（3）测试方面要做好Smoke Test。在开发人员做完修改后，做代码Check in之前，做一次Smoke Test，能及时发现问题。按照以前的安排，我们每天只在早上做一次BVT，后来针对这种情况我们在下午6点增加了一次BVT。下午的BVT就相当于做Smoke Test，尽量减少第二天凌晨Official Build不能正常工作的情况的出现。

在等待一个稳定的build的几天里，我因祸得福，利用这段时间增加了一些Test Case，增加了测试的覆盖率。

在项目进入胶着期时，最忌讳的就是互相指责。任何指责的话语都会导致矛盾激化，可能会瓦解一个团队。不管是管理者还是普通员工，这个时候都要管住自己的嘴。只有团结一致，团队才能迅速度过胶着期。

不要在不具体的问题上争论

微软内部分为很多层次（或者说等级），有很多高水平的人在做管理，观察他们的言行举止，就能发现一些值得学习的地方。

一次开项目的总结会，在北京的测试人员和开发人员全体参加，这个项目的Manager也应邀参加。因为级别比较高，平常他不参加我们的会议。

当时一位测试人员与一位开发人员在一件事情上产生了争执，他听了听后，插话进来，给了他们也是给整个项目组3点建议：

（1）不要在不具体的问题上去争论。这么High lever（高级别）地争论问题没有意义，不如回去两个人坐下来，翻翻材料，拿出具体的证据来。

（2）这次总结出来的问题，有没有是在上次总结会上就已经总结出来的？如果有，那么为什么这个问题会再次出现？为什么上次我们就说要改进而这次又说同样的话？

（3）总结出问题，很好。具体的措施在哪里？怎么去做？

听到这样的发言，你能发现他的眼光很犀利，直接就看到了本质。他的问题虽然不好回答，但只要回答了，工作效率也就有了提高。

改进测试工具的开发流程

之所以单独把这个话题提出来，是因为测试工具的开发有一些的特殊性：

（1）测试工具的开发依赖于产品。随着产品的改进，测试工具也需要同步改进。Server端的测试工具依赖Server端的协议，协议变了，测试工具必须马上更新。Client端的测试工具一般依赖于界面元素的ID，ID变了，测试工具也要更新。

（2）时间紧。编写测试工具只是测试人员工作中很少的一部分。

（3）没有人来做针对测试工具的测试。如果不是人力特别充裕，一般的项目组里都不会把对测试工具的测试作为一项正式的工作，其测试由工具的开发者（测试人员）兼做。

针对这些特点，我的建议是：

（1）和开发小组进行协调，尽早地稳定Server端的协议和Client端的Element ID（元素的ID号），且不要轻易更改。一旦发生变化，也要及时通知测试组，让测试组提前做好测试工具的更新。

（2）把测试工具的编写作为一个正式的工作任务分配下去，并保证不占用太多时间，而不是让测试工具的编写变成一项“隐含”的工作。

（3）安排非编写者来做简单的测试。因为测试组的工作是测试产品，不会有很多的精力来测试测试工具，但简单的测试还是有必要的。测试工具的不准确势必导致测试结果的不准确。

从用户反馈报告和bug中学习

因为在M1只是搭建了一个稳定的产品构架，实现了基本功能，所以我们的“丑媳妇”开始见“公婆”是在M2结束时。M2结束后，微软公司选择了10名客户，由他们来试用我们的产品，并提供反馈。

当陆续收到转发过来的用户反馈的邮件时，我感到了震惊。因为我负责UI测试，用户接触最多又容易发现问题的也是UI以及其方便性，所以这方面的反馈要多一些。测试人员擅长发现在特别场景下出现的功能错误，

而最终用户擅长发现一些普通场景下的UI错误或者使用性上的不便。下面列举两个例子：

（1）在一个对话框的标题栏里，我们把一个单词中的两个字母换了一下位置，写了一个不存在的单词☺。这个错误是简单的，而没有发现又是遗憾的。我们这些测试员看过这个对话框千百遍，却都没有发现☺。

（2）用户觉得很多窗口的宽度都略微小了一点，不方便使用。这一点也是我们这些一心钻到产品里面的人看不出来的。

所以我想，看用户反馈报告是一个很好的学习机会，我们可以从一个旁观者的角度来审视自己以及整个小组的测试思想，能够从中得到一些很好的想法，有了想法就会有新的测试用例。

另外，我觉得，我们也可以从已发现的bug中学习。虽然测试都是依照测试用例来做，但是，实际的测试工作中都会对测试用例做一些延伸，例如，测试用例中要求是点击一次按钮，你可能会在得到正确输出后再试试，随意点击多次，或做点别的。这样，我们通常能够发现一些测试用例没有捕捉到的bug，因为毕竟测试用例的集合只是一个主干，它不可能覆盖所有的情况。一段时间后，这些“顺带”被发现的bug就会有些积累，我们可以拿出一个小时，回过头来看这些bug，不管是自己发现的还是别人发现的，分析为什么测试用例没有覆盖它们。分析结果出来了，新的测试用例也就有了，这样我们的测试用例集合会更强壮。

通过对用户反馈报告和bug的学习，我们能够提高测试覆盖率。

Before filing a bug, just repro it again

（提交bug前，请先再次重现它）

开发人员和测试人员在工作上有些争论是很正常的，而有些事情却容易加深误解，虚假的bug就属其一。对于一个bug，开发人员花费时间去检

查代码，最后却发现这是一个假bug，自然怒火中烧。测试人员得知消息后也觉得很无辜，争辩说，我不知道啊，我的确……

为什么开发人员不适合做测试？那是因为开发人员总是有一种心理预期，“我的程序没问题”或者是“一点点小的问题，没有什么，简直可以忽略不计。”。在这种心理预期下，开发人员不容易发现问题，而在测试人员总是怀疑有bug的心理倾向下，我们要预防一种错误：报告虚假bug。测试人员一旦发现问题，心里就兴奋起来，这种兴奋程度与bug的严重性成正比。兴奋起来以后，人的理智便下降了，人的所见所闻可能与实际情况就有所差异。例如，很有可能是因为测试人员没有执行规定的步骤，导致了预期外的结果，而测试人员被这个错误的结果迷惑了，兴高采烈地把它当作bug提交上去了。

如何避免这种情况呢？

我的建议是，在测试的过程中发现了问题，暂时不要作为bug提交，而是概要式地记录在一个文件里或者一张纸上，然后继续往后做测试。几个小时后，测试工作告一段落，我们可以做一个短暂的休息，放松一下大脑，然后再回过头来，处理这段时间内发现的问题。一个一个地重新做一遍，再次发现的问题，就是真正的bug。测试人员不是神明，也是普通的人，也会犯错误，我们要给自己一个审查自己的机会，以避免虚假的bug。

在提交bug前，查询一下bug库或者和相关的同事做一个简短的交流，尽量不提交重复的bug，这也是一个良好的习惯。

专业，这个词的分量实际是很重的。虚假和重复的bug越少，我们就是把测试工作做得越专业了。

为什么在以前没有发现这个bug

测试人员最怕别人问什么？我想，测试人员最怕的问题是，当他（她）

提交了一个bug后，领导或者开发人员开始追问：为什么在以前没有发现这个bug？

这个问题好难回答。当然不能说，我不知道，我不对以前没有发现负责。这样的回答就会引起争吵，就会不欢而散。如果大家去掉指责的态度，都心平气和地坐下来分析，就会发现，一个bug在以前没有被发现可能是因为很多种情况。不同情况要不同对待。

一种是新功能的bug。这种bug当然不可能在以前被发现，因为新功能刚刚实现，对这一块的测试也刚开始。

一种是测试用例不完善造成的。有的是测试用例覆盖率不够，以前没有想到，这次偶然抓到的。有的是虽然有相关的测试用例，但是并没有详细的测试数据，测试的随意性很大，导致以前漏掉了。这个时候，需要测试人员诚恳地低下头来，去更新测试计划和测试用例，确定具体的测试数据，提高测试覆盖率。

另一种是开发人员在修改代码时引入的新的错误，术语上叫Regression bug。出现这种情况的时候，容易发生争执。开发人员说这个错误以前就存在，测试人员疏忽了，没有发现。测试人员说，这个功能以前是正确的，现在被改错了，是Regression。争到后来，就看谁的底气足，谁有证据了。

在前面“胶着”这一篇内容里，我提醒大家不要指责，在这里，我要对开发人员重复唠叨一下。我们国家的文化的特别之处在于，大家还是很在乎旁人的认同感的，也比较喜欢与谦虚的人相处。如果“为什么在以前没有发现这个bug”这个问题出自于开发人员之口，一是抹去了测试人员的功劳，二是显得开发人员不谦虚。一个bug发现得晚，自然有Test Leader去问责，不管怎么样，Bug的出现毕竟和开发人员有关，如此责问，怕是不能服人。

测试人员遇到这种问责的时候，好像一根鱼刺卡在喉咙里，上下不得。但虽然难受，却也是提高的好时机。在大的压力下，动力也大。话虽然难

听了一点，但让我们看到自己的不足之处，进而改进了测试计划和测试用例，因此只要不带人身攻击，只是单纯的工作上的质询，又有什么不可接受的呢？

Ask questions, please（请多问问题）

在M1结束后，美国的同事带来一个反馈，就是美国的同事觉得在开会的时候中国这边太安静了，很少有人主动发言。他们的疑问是，是真的没有问题还是没有听懂？

在Beta结束后，美国的同事们再次来到中国访问。因为美国那边做了机构调整，有一些同事刚刚参与到这个项目中来，包括大老板☺。项目组就做了一次产品的体系结构的Review，就是把整个产品的体系结构讲给大家听，一是为介绍，二是为审查。整个会议上提问的基本都是美国的同事，而且很频繁，不一会儿就能听到“Excuse me”，之后马上接着问问题。

一天，美国方面的测试Leader给我们做了一个演讲，他要给我们共享的第一个经验就是：“Ask questions, please”。

他说，在你真正懂得一件事情之前，不要轻易地去估算它。例如一个新的功能，如果你不了解它，你不会知道为它你要花多少时间。在给出你的测试时间估算之前，你要对Spec中任何有疑问的地方提出你的问题，同时考虑你需要编写的测试工具及其需要的时间。

我们中国的文化趋向于内敛，当我们要提出一个问题的时候，先天地会有两个问题在头脑里打转：

（1）这个问题提出来后，主讲者会不会不知道怎么回答？他会不会很难堪？

（2）这个问题提出来后，会不会太简单？会不会显得自己比较愚蠢？

经过这一番自动“筛选”，问题所剩不多。加上本来一个问题在会议

中的“保鲜期”就很短，等你鼓足勇气提问的时候，主讲者已经进行到下一个议题，你只好作罢。

文化这个议题太大，在这里谈不免是空谈。可是，如果你是一个管理者，你就需要细思量，开会是为了什么？沟通。沟通的最便捷的途径当然是提出问题和解答问题。如果大家有疑问而不说，沟通的效果就没有达到，会议的预期目的就没有达到。换句话说，公司浪费了钱，成本增加了。我们可以考虑从这两方面去改进：

（1）会议要有一个平等和谐的气氛，让与会者提问时没有什么顾虑，主讲者也不会认为别人在存心刁难自己。如果主讲者来自管理层，就更要注意，不要威风凛凛。

（2）要鼓励提问。在每一小节结束后，都留有问问题的时间，使得大家有机会从从容容地问问题。

我怎么能够帮助你提高，我的员工

因为经济增长的速度赶不上劳动力的增长速度，现在最火爆的场面就是招聘会了，要想进入微软这样的大型跨国公司比考大学、考研究生难多了。那么，挤进了微软之后是一种什么样的情形呢？微软对自己的员工会不会只让干活而不考虑员工的提高呢？反正想进微软的人多得是。事实恰恰相反。

微软很重视自己员工的提高，并将其提供了多种方式和便利条件：

（1）正式培训。微软有专职的培训讲师，为各种员工提供培训，包括技术、销售、财务、人力资源等等。参加培训是工作的一部分，很多培训就安排在工作时间，你不必为参加培训而牺牲自己的业余时间。

（2）员工交流。如果你是微软的正式员工，你肯定有机会到美国去工作一段时间。如果你是美国员工，对中国文化很感兴趣，你可以申请到中

国工作。在先进技术的保障下，一个员工工作的物理位置的变化不会影响到他（她）的工作效果。但，这个物理位置的变化对员工却意义重大。另外，一个项目如果是跨国合作的，双方都有机会到对方的办公地点去做工作访问。毫无疑问，这种跨国的员工交流是很受欢迎的。

（3）讲座。公司鼓励在某一方面有深入研究的员工把自己的经验总结出来，给大家开讲座。开讲座，一方面满足了大家需要提高的要求，另一方面也给大家提供了一个展示自己的舞台，一举两得。这种形式的培训成本低，公司只需要提供会议场所就可以了。

（4）鼓励学习英语。因为英语是微软的工作语言，所以大家都很关注它。ATC（微软亚洲工程院）把周二改为英语日，鼓励大家使用英语交流。这样做的初衷是要去掉大家都不好意思说英语的害羞心理。我所在的项目组因为项目是跨国合作的，使用英语的场合会更多，ATC还特意从美国请了一位vendor来提高我们小组的书面和口头英语的能力。他参加我们的会议，审阅我们的文档，切实提高了我们的英语水平。

（5）图书馆。微软的图书馆有两种形式，一种是普通意义上的图书馆，有一间专门的办公室，你可以去借阅书籍。另一种是数字图书馆，你可以找到很多有用的资料。有一次我在内部图书馆的网站上借了一本书，当时没注意看，等接到通知去领快件时才发现是从美国总部图书馆寄过来的。

（6）订阅了多种期刊杂志。这些杂志就摆在休息室内，随手可得。

看过这些，不知道你是否已经看到微软巨人正在弯下腰对每一位员工说：我怎么能够帮助你提高，我的员工？

休假

前些天，在美国NBA新闻中看到这样的一则新闻，说一个球员只打了上半场，下半场因为老婆要生孩子赶回医院去了，导致球队少了一位主力。

我相信很多人对此都会感到诧异。这的确可以看出各国文化的差异，这种事情在我们国家发生的概率太小了。我们的文化鼓励每个人讲奉献，克己奉公，忠孝难两全时以忠为大，舍小家顾大家，等等。在这里我并没有批评中国传统观念的意思，只是想介绍一下另外一种观点。从我看来，两种都可接受。

在项目的进程中，我们发现，美国同事的休假有三个特点：一是国家规定的正式假日一般都和周末连在一起，这样就有了一个三天的小假期，不会像我们一样把两个周末调到一起来凑成一个大的假日；二是休假频率高，他们通常会在周末后面加休几天，这样一年的带薪假期就分成了多次来休；三是休假的时间选择是依据自己的需要，而不是依据工作的进展情况。一般来说，在我们中国员工的心里都有一个尺度："等到项目比较空闲的时候再休假。"，这点是不需要老板说的，而美国员工不是这样，项目再忙，也会休假。

参与到跨国项目中，能够看到两种不同的文化，挺有意思的。所以，我很鼓励大家来做外包，这将会是一个打开个人眼界的宝贵经历。

IT服务

在微软有很多的计算机高手，那么微软是否还需要网管、系统管理员这样的角色吗？大家遇到什么问题自己解决不就行了吗？事实上，微软有一个庞大的IT服务体系，给大家提供服务。

在北京知春路的希格码大厦的微软办公地点，如果你在计算机系统方面有任何需要帮助的问题，你可以拨打电话6000。微软为什么会维护一支IT服务队伍呢？我想这也是优化的结果。

微软的内部办公网络很庞大，体系复杂，肯定要有一个部门来做专门管理。那么，有什么必要对员工也提供一些日常问题的技术支持呢？

首先，任何一件事情，如果由专业人士来做，效率肯定高得多。例如操作系统的安装，的确简单，大家都可以做，但IT服务人员通过特定工具来做系统恢复，很快就能做完。

其次，给大家节省了时间。例如你在做一个项目，一天一台计算机有点问题，需要重新安装操作系统，或者硬件出了问题系统不能启动，这时你可以把它交给IT服务人员，你则可以继续做自己的事情。等计算机送回来了，你又可以使用这台计算机工作了。节省了时间，也就是节约了钱。

再次，有利于问题的迅速解决。IT服务部门会把遇到的问题总结出来，给每一个服务人员提供参考，经验共享，这样解决问题的周期会大大缩短。计算机的分支很多，问题更是五花八门，员工再厉害，也只会对某一领域熟悉，相对来说，专门解决问题的IT服务人员见多识广，问题解决起来要快得多。

因为微软内部有一些规定，例如解决问题的周期，响应时间，解决问题后由员工填写意见调查表等等，这使得员工真的成了IT服务部门的客户，能够享受到很好的服务。这一点也是挺重要的，不然的话，如果养了一群大爷做IT服务，响应特慢，服务冷淡，那真不如没有。

让软件更好用

看到过一些评论，说其实Windows 在技术上没有Mac或者其他操作系统强……。我没有能力在操作系统的层面上说孰强孰弱，作为一个普通用户，我还是会毫不犹豫地选择Windows，因为它好用啊。

易用是微软生产软件的一项重要指标，微软的软件的易用性主要体现在下面几个方面：

1. 界面设计时考虑方便用户。

（1）界面简单；

（2）控件标准化；

（3）智能，帮用户记住需要记住的事情；

（4）不同窗口之间的逻辑关系简单，不会把用户弄蒙了；

（5）界面上尽可能提供帮助文档的链接，例如很多界面上都有“帮助”按钮。

2. 随同软件一起发布的，还有很多帮助文档，例如：

（1）软件简介：介绍这款软件的主要功能和适用对象；

（2）软件安装文档：详细介绍如何才能安装和部署这款软件；

（3）帮助文档：详细介绍软件的每一个功能，每一个流程。

3. 在网上提供常见问题列表和解决办法。

4. 提供人工技术支持。

虽然我们国内很多软件公司生产的都是行业应用软件，但我们仍然可以学习微软在易用性方面的做法。的确，要做好界面设计、编写各种文档，都会增加软件的成本，但我们可以先有这个意识，从简单做起。软件不分大小，总会有客户在使用，多一个客户觉得方便使用，软件就会成功一分。

办公区的N个细节

有的人说“细节决定成败”，有的人说“做大事者不拘小节”，这些都是各执一词的说法。我觉得在微软办公区有一些有意思的细节，和大家分享一下。

1. 贴在会议室门上的纸

在每个会议室的门上都有一张小纸，上面打印了这个会议室今天的使用情况。因为微软的会议室都很忙，如果你想临时找一个会议室讨论一些问题，看这个小告示就知道了会议室的使用情况，免得被人在会议的中途赶出来。

2. 内部张贴宣传画

相信很多公司都有自己的宣传画、宣传单，那么你是否想过把这些东西张贴在自己公司的办公室里？微软是这样做的，这样，员工也就知道了最近公司在做什么活动。

3. 电梯口的 Office 告示

现在做电梯口的电视广告的分众媒体已经赚了大钱了，微软同样也没有放过电梯口。在希格码大厦，如果是微软的专用办公区，你会在电梯口发现教你一些Office使用技巧的小告示，很有意思。反正等电梯也没事，看看，还能学点东西。

4. 会议室供应茶水

开会并是一件严肃的事情，一般会议开始前服务员会送一壶茶水来，桌子上有纸杯，可以随便饮用。如果服务员没有送来，你可以打电话要，一会儿就送来了。有的会议还会提供点吃的，例如饼干之类的。如果是中午午餐时间开会，就会提供汉堡之类的快餐，让你吃饭、开会两不误。

5. 处处能找到系统管理员的联系方式

在投影仪上、打印机上、电话上，都张贴有系统管理员的联系电话，一旦这些设备出现问题，你能很快找到他们。很方便，缩短了问题解决的时间。

6. 可以去领方便食品

如果已经是下班时间而你在继续工作，你觉得饥肠辘辘的话可以到前台去领一点方便食品，例如饼干、方便面，可以帮你撑一段时间。

7. 无处不再的黑板

每个会议室里都有大黑板，老板的办公室里也必有黑板，休息室里有黑板，甚至休息室的桌面也是黑板，同时都配有水笔，你想讨论什么你就随便画吧。

8. 办公区的花

在办公区的边边角角，摆有很多花。工作累了，站起来，看看它们，有的时候还忍不住摸一下它们绿绿的叶子，挺有意思的。

9. 每个办公区都有厨房

厨房里有冰箱、微波炉，提供饮料、咖啡等。

10. 利用墙壁办展览

墙壁上会有一些展览，例如外出旅游的照片，某个项目的庆祝晚会，公司的活动照片，这些会吸引路过的人停下来观看，反正墙壁闲着也是闲着。

公司无论大小，都可以在一些细节上多做改进，花费不多，说不定却很有成效。

权限控制

像微软这样的大公司，内部网络特别庞大，权限控制也很复杂，复杂性主要体现在：

（1）微软的员工分为多个级别，权限各不一样。

（2）有很多部门，有很多项目组。

（3）有在内部工作的vendor和在公司外部工作的vendor，他们都需要访问内部网络。

（4）除了美国总部以外，微软在多个国家和地区设立了分支机构，这些机构之间也需要互联互通。

（5）员工在旅途中和在家里也需要接入公司网络。

如果没有权限控制，微软的知识产权就成了泡沫，顷刻间灰飞烟灭。

著名的XBOX项目曾经有一小部分在北京开发，微软公司给他们小组单独配了一间会议室，以和大家分隔开来。会议室的玻璃墙上从内部全部用

海报纸贴上，一直贴到天花板，远看就像一个纸盒子。门上白纸黑字地贴着：XBOX（绝密）。

这也是在提醒大家，非相关人员，请勿靠近。

我是属于在微软内部工作的vendor，我可以访问很多资源，但也有很多不能访问。当我访问到没有权限的资源时，系统会提示，这个资源你无权访问，或者如果你同意的话，它会自动给你的老板发信，确认你的确需要访问。这样，既给工作提供了方便，也限制了可能的非法访问，保护了公司的利益。

我接触过的一些公司，有的可能是没有在意，或者技术上无力实现，内部网络基本上属于无权限控制状态。虽然有服务器，但对各工作机却没有控制功能，整个公司只是分成了几个工作组而已。在这种情况下，如果连接上互联网，没有任何安全可言。即使没有互联网接入，在资源共享的情况下，一个移动硬盘就能把公司产品的源代码全部拿走，让人痛心。

作为一个公司的管理者，我们既要给员工创造良好的工作条件，也要保护好公司的知识产权。没有了知识产权，公司就没有了核心竞争力。给员工规定合适的权限，并把这个权限集中控制在少数的高级管理人员手中，能够保护公司的利益。普通员工也应当对这种严格的权限控制予以理解，只有公司的核心利益得到保护而不被少数人非法盗取，员工的利益才能得到保障。不然的话，公司垮了，什么都是白说。

另举一个例子，作为vendor，我们没有权力带亲戚或者朋友进入微软办公区。当你尝试着这么做的时候，保安会坚持要看你的门卡，如果你只是vendor，会遭到拒绝。虽然执行起来有点不近人情，但也是没有办法的事情。微软的保安都很精干，与工作人员没有什么联系，执行起来很是不折不扣，没有什么情面可讲。

限于我的学识，我觉得架设一个用严格权限控制的公司网络来说，微软公司的Active Directory是一个很好的产品（就是平常所说的域的概念）。

测试推动研发

我知道现在有一个名词是“测试驱动开发”，那是指一种编码方法，我这里的“测试推动研发”是指测试会在某种程度上成为推动研发工作的一股力量。

我们知道，一般来说，测试是一种被动角色。一项功能，最先是PM定义，然后是开发人员实现，最后才是测试人员做验证。测试如何变被动为主动呢？先说一个实例吧。

在我们的测试小组中，由同事YD负责测试电话号码的设置。在项目初期，可能是因为PM想得简单了，对这个功能只是概括性地做了定义。YD在做测试的时候发现，有很多细节在需求规格说明书中没有定义，这样导致了开发人员的处理简单，从而滋生了软件错误。她在报告产品bug的同时，积极地写邮件陈述需求规格说明书中需要提高的地方，推动PM去改进。电话号码设置只是我们产品的一个小功能，所以推动起来不大顺当。但是YD持续地去做，包括写邮件、报告bug、在各种会议上提出来，最终，这个功能得到了很大的提高，整个研发小组都认为YD对这个功能起了很大的推动作用。后来，YD得到一个称号——Telephone number queen（电话号码皇后）。这既是一种开玩笑，也是对她工作的肯定。

发现和报告bug是测试工程师的本职工作，只要把这个做好，没有人可以责问其他。在这个例子里，难能可贵的是YD能够本着提高整个产品品质的思想出发，“多管闲事”，推动了整个研发工作。我相信，任何一位管理者，都希望自己的员工能有这样的主动性。在这里起关键作用的，是YD的专业素质。

微软有一种很好的工作氛围，总是鼓励大家做得更专业。YD做事认真，是一个专业的测试工程师，所以她在默默地推动着电话号码设置这个功能的研发。

作为测试工程师，我们的主要任务是发现bug，但如果项目实际情况需要我们去推动研发工作，我们就要自动“补位”，迎难而上。只要是有利于项目的事情，我们都可以去做，因为只有项目成功了，个人才有可能获得成功。

开发人员做代码变动需要得到批准

从理论上来讲，开发和测试是平等的，这个大家也都能接受，只有平等才能进行有效监督嘛。可是，“泥巴怎么捏”的权利在开发人员手里，他在大家都不知情的情况下对代码做改动，你能怎么样？还能找他打一架？

在项目后期，不管开发人员出于好意还是出于对技术或者完美性的追求所做的代码变动，实际上都是项目的一个风险。很有可能他的改动中包含错误，也有可能他的改动会对其他模块带来不利影响，他不可能考虑得那么全面，所以，我们需要控制这种改动。从配置管理的角度来说，对变更的控制是项目的一个关键。项目必须在控制之中，而不能被某一个开发人员的“心血来潮”拖入泥潭。

微软这方面做得很好。到了项目后期，每一个代码的变动都需要经过评审小组的批准，没有得到批准而对代码做变动会被公开批评。在一个专业的小组中，被公开批评是一个很大的负面评价。这个评审小组的成员包括开发组长、测试组长、需求组长等人，他们每天开会来决定当前有哪些变动可以被批准。所以，你会看到，开发人员会小心地陈述理由以争取变动获得批准，他们的“随意修改”的权利已经被剥夺了。有的时候，当的确是因为编程上的错误导致了一个软件错误时，开发人员想修正自己的错误，在争取批准的时候就会比较着急，生怕得不到批准。

这个制度巩固了开发和测试的平等关系，避免了产品质量的动荡，值

得学习。

认真考虑用户的真实使用场景

我在项目组内的工作之一就是测试Toast，我是这个功能的Owner（负责人）。当你使用MSN的时候，你的朋友发消息给你，你会在屏幕的右下方看到一个小窗口，这就是Toast。我们的产品虽然不是MSN，但是是类似产品。

Toast虽小，可是我不敢含糊。从Toast的种类、触发条件、显示内容、UI效果等出发，我编写了多个测试用例。几轮测试下来，自我感觉还不错。

可是在测试后期Beta用户报告的一个问题却让我有些惭愧。这个问题是这样的，的确，每次有联系人登录或者收到一次对话的第一条消息的时候，用户都希望看到Toast，这样用户能得到明确的提示。但是，从另外一个方面来讲，用户也不希望当前的工作被中断，例如，如果用户在写邮件，他既希望看到Toast，也希望能够继续写邮件，用户的当前焦点不能变。这就要求Toast显示出来后，既要显示到桌面的最前，又不能抢夺用户的焦点。这个问题的价值很大，它让我看到自己的测试思维有一个盲区：用户的工作环境。这个工作环境不但包括用户的计算机硬件、操作系统、办公软件、浏览器，还包括工作场景（scenario），就是用户的某一时间段的工作状态。

在测试环境上，测试总有一个要求：单独的、干净的测试环境。这是没有错的，只有在单独的干净的测试环境中得到结果才有意义。但是，这并不表示我们不需要考虑用户的实际工作环境。从另外一个角度来看，我们还要考虑用户可能在什么场景下使用我们的产品，要设计出这方面的测试用例，以发现可能存在的问题。我们照样可以在测试环境中模拟出用户的办公场景，这与环境的独立性和干净没有冲突。

这样的考虑是不是有点多余？如果你现在的产品并不是商业软件，只

卖给一个或者几个特定的用户（为讨论方便，我们称它为项目软件吧），或许你会有这样的想法。其实一点都不多余，项目软件的领域中也存在着竞争，项目软件的用户也会越来越挑剔，如果你能够为用户多考虑一点，就能为你的产品博得好的声誉，在竞争者当中脱颖而出。在竞争公司技术水平大体相当的情况下，这种胜出是由类似的多个细节组成的。

Non-goal（非目标）

在软件这个行当里，自然少不了各种各样的文档，例如需求规格说明书、概要设计、详细设计、测试计划、测试报告，等等。在这些文档里，一般都会有一个小节来叙述产品或者本文档的“目标”，例如需求规格说明书的这个小节里简明扼要地列出产品的主要功能和客户群。

我在微软工作了一段时间以后，接触到一些文档，发现了一个有意思的细节，这些文档一般都有一个non-goal的章节，用以说明软件产品所不会实现的功能或者不在本文档讨论范围之内的事情。例如，你正在编写一个字处理软件的需求规格说明书，你觉得手写识别这个功能好是好，但限于技术能力，是你这个小公司无法实现的，你就可以把手写识别功能写到non-goal中。再例如，你在编写一个产品的测试计划，你觉得这个产品的性能测试很复杂，需要另写一个测试计划，你就可以在non-goal中说明，性能测试不在本文档覆盖范围之内。

当时我只是觉得有趣，因为很少见到这样的陈述。慢慢地，我觉出它的好来了。任何一个软件产品，必定有它的功能边界，它只能实现一定的功能，不可能什么都可以做，non-goal可以打消一些不切实际的期望。另外，一个文档也只会侧重在某一方面，清楚地描述出不在本文档讨论范围之内的事情，也可以分清文档之间的边界，节省阅读时间。

可能是因为计算机的威力太大了，有的时候它简直就是无所不能，这

也给一些客户造成了一个假相，似乎只要他们张嘴，新需求就一定能实现。在国内软件业竞争激烈的背景下，客户可以不管软件开发的实际阶段（可能已经到了最后一个里程碑或者最后几天），不管需求是否符合实际（可能根本用不上），不管软件公司是否能够承受（可能软件公司已经不可能再做软件基本结构的修改了），对软件提出不可抗拒的修改意见，有的修改意见甚至就好像楼房已经建到10层，来了一个修改地基的需求一样。我知道，即使在需求规格说明书中增加non-goal，对这种情形也并没有什么作用，但或许对客户有一个心理作用吧，软件产品也只是普通的产品而已，不是橡皮泥，能捏成任何形状。

同事之间的审阅

如何提高代码质量，是一个经常被提及的话题。测试是软件质量的一道防线，是不可缺少的，而且是软件开发“三分天下”中的支柱之一。但是，如果能提高软件代码质量，减少bug总量，也就减少了软件的风险。

软件开发因为其工作的特殊性，很难进行管理。我的职位虽然不是开发，但我经常能收到开发人员提交代码的通知邮件，在这类邮件里，我注意到，都有一栏：审阅者（Reviewer）。也就是说，如果你是一个开发人员，在你向代码服务器提交代码之前，需要找另一位开发人员来做审阅，在取得他（她）的同意的情况下，才可以提交代码。如果发现了错误或者不完善的地方，应当立即进行修改，再做审阅。

我觉得这样做的好处有：

（1）增强了开发工作的透明度。

（2）在具体的开发细节上都体现了多个人的智慧。

（3）这是切实可行的监督方法之一。如果等产品都开发完了，再来审阅几万、十几万行的代码，简直就是不可完成的任务。

我认识一位建筑业的驻地监理工程师，我向他请教了他们是如何做监理的。建筑业的风险也很大，代码有的时候还能推倒重来，建筑是不可能了。他给我举了一个例子，他说，有一次监理一个立交桥，他们的工作仔细到对每一车的混凝土都做检查，并且都要留样。所以，建筑工人工作到什么时候，他们就要工作到什么时候。我想，这种严格的精神也需要在软件业得到承传。

同样，这种同事审阅的制度也可以迁移到测试上。如果一位测试工程师修改了测试工具、测试计划或者测试用例，也需要找另一位测试工程师来做审阅，然后才能提交，我相信这会提高测试工作的质量。

Dog food（bug悬赏）

当我第一次看到这个名词的时候，不禁哑然失笑，心想，还真贴切。

一般当一个产品做到RC（Release Can，可发布的候选版本）阶段的时候，微软公司会安排一次Dog food，就是公开悬赏，谁抓到bug谁拿奖。

下面是一则模拟的Dog food布告（通过邮件发布）：

各位同事：

大家好！

X产品是一款……（产品介绍）。

欢迎大家参与X产品的Dog food活动，你可以通过访问指定地址来下载安装。发现的bug请发邮件给YY，我们会及时答复你。活动从今日开始，到月底结束。结束后，我们将评出金、银、铜三个奖项：

金奖一名，奖品是MP3一个

银奖两名，奖品是高级耳机一副

铜奖三名，奖品是鼠标一个

感谢你的参与！

所谓“重奖之下必有勇夫”，这些奖品虽然不是“重奖”，但很有意义，给了大家一个比试测试能力的一个小舞台，而且，大家通过参与这项活动还能了解一款新产品，所以还是会有很多人参加。参加的人越多，对Dog food活动的主办小组就越有利，越是发现了大问题，收获就越大，因为不管怎么说，都是在公司内部发现的，这比客户发现它要强一百倍。

本项目的人不参加Dog food，以示公平。当然，要得这个奖也很不容易，因为到了RC阶段后产品都很稳定了，要想发现一个bug得下功夫。

通过一些小小的奖励来鼓励项目组之外的人来找bug，是在另外一个层面上调动了更大的资源，是一个投资回报率很高的活动。Dog food活动的一个副产品是在公司内部为这个产品的项目组做了宣传，扩大了知名度。

这种方式即使是小型软件公司也可以效仿，简单易行。

RTM（Release To Manufacture，产品发布）

经过中美两个小组一年半的奋斗，我们终于迎来了产品RTM的日子。

邻近RTM的前一段时间里，气氛很紧张，任何一个问题，都会引起高度重视，大家确认再确认。首先要确认这是不是一个问题，可不能拉响假的空袭警报啊。如果的确是问题，它的严重性有多大？是不是一定要做修改呢？修改的成本有多大？此时开发人员做代码修改和和测试人员做验证，都是战战兢兢的，生怕引起什么其他问题。

如果要做一个比方的话，我觉得，产品小组成员对RTM的期待就好像是老父老母在年底对外出工作一年的孩子回家过年的期待。在RTM那一天，当然开发和测试工作都停了下来。大家脸上暗藏着笑容，做着一些零散的工作，整理整理计算机的磁盘，查看查看邮件，……有的时候两个人之间还有一些简短的对话：

“嘿，今天能RTM吗？”

“不知道呢，应当会吧。”

“但愿。”

“是啊，但愿，都望眼欲穿了，哈哈。”

是啊，虽然今天是RTM，可是在没有得到正式的官方消息之前，谁能肯定呢？

下午四点多，老板拿来两张大纸，上面写着：CWA1.0 Sign Off，让我们每个人签字，表示同意发布。之所以要准备两份，是因为要给美国的同事留一份保存。这个签字的过程还是很激动的，办公室里一下子就热闹起来。

签完字后，大家一起去前台照相。微软亚洲工程院院长和我们项目的高级经理BG也赶来祝贺，和大家一起合影留念。我们把产品名称打印在几张大纸上，由几个同事举着，很有意思。合影过后，同事们纷纷在微软亚洲工程院的牌匾下单独留影，以做留念。

因为照相的时候已经是下班时间，微软其他员工下班经过前台的时候看到我们那么高兴，禁不住多看几眼，我想，他们肯定很羡慕我们，也会祝贺我们的。

晚上大家一起出去吃饭，欢庆RTM，欢庆胜利。

有的同事在MSN上留言，I love CWA。项目能做到这样的程度，真是让人欣慰。

诚然，作为中小型软件公司来说，我们的产品大多是针对某个客户的项目产品，不需要RTM，但是，在项目结束开发的时候，也应当有这样一个仪式，它同样是一个重要的里程碑，对大家的心理影响是深刻的。这样的仪式不会花什么钱，但会让大家感觉到成就感，感觉到一个任务的完成，之后也会更加紧密地团结在一起。

缺点什么

在这里我并不是要说微软缺点什么，微软它自己会思考，我要问的是自己。

封闭是一种自然状态，特别是在紧张的项目组里。日常打交道的都是组里的同事，谈论的话题都是与项目相关的，在大家越来越紧密地联系在一起的同时，与外界的交往也就少了。可能你也会同意，在保持对本项目的高度重视的同时，有的时候还是要走出去与外界交流，看看别人是怎么做的，他山之石往往很有启发意义。

我们的产品发布后，我和项目组的几个同事听了一个讲座，是一位来自Exchange的一个项目组的工程师介绍他们的自动化测试方案。因为我们产品的服务器端的管理程序和Exchange的类似，都是基于MMC（Microsoft Management Console，微软管理控制台）的，虽然具体功能相差很大，但在测试方面有很多共通之处。听完讲座后我们都觉得Exchange项目组做得很棒，我们项目组负责MMC测试的同事Sf更是觉得很受启发，决定开发一套测试平台，提高服务器端的测试自动化率。

微软很庞大，在希格玛大厦就有微软的好几个部门，像微软亚洲研究院、亚洲工程院、中国技术中心，等等。我们这些来自同一家vendor公司的员工分布在不同的楼层，平时很少见面。一天中午，我约了两个同事一起吃饭，饭后我到他们办公区看了看，觉得挺有意思的。他们是做游戏测试的，他们所采用的测试方法是我闻所未闻的，使我开了眼界。

我曾经看过一个书名，叫《不要一个人单独吃饭》，作者的意思是要尽可能地约人一起吃饭，利用一切机会交流。书名是有点夸大其辞，这样才会引起读者的注意☺，但表达的意思是对的。

有很多资源，实际上是自己走出去一步就可以获得的，而如果不在那个环境里，想得到这些资源难度是很大的。例如，如果不在微软工作，不

可能听到内部技术讲座，也很难看到其他项目的测试情况。另外，和不同项目组的人聊天、交流，会给我们带来不同的思路，开阔眼界。种种此类资源，不论公司大小，都是会有的，只是需要我们去发现，。技术工作有封闭的倾向，但只要意识到了，主动走出去，是很容易克服的。

最大的收获——专业精神

可能有人要问我，在微软工作的这一段时间里，你最大的收获是什么呢？这是一个很好的问题，我要认真想一想。

我觉得我很幸运，到微软后就赶上了一个好项目。这个项目在微软来说很小，却是“麻雀虽小，五脏俱全”，其包含了多种软件技术。我从M1就进入到了项目组，一直到最后产品的发布，全程经历了一个完整的也是成熟的软件开发流程，这种项目经验会让我受益很大。

然而，我仍然觉得，我在微软工作的最大的收获是见到了一种专业精神，并受到了些许熏陶。

微软的工作氛围是很好的，无论是在中国还是美国总部，有很多优秀的员工，无声地做出了榜样。从邮件书写，谈话方式和语气，表达自己的观点，到对bug的态度，对变化的应对，对新东西的理解等等，从中都能发现专业之道。大家都是在努力地把事情做好，没有什么杂事搀和在里面。例如，前几天你对一个问题的看法错了，你会发一个邮件承认自己错了；开会迟到了或者错过了会议，你会主动发邮件告诉大家原因并表示歉意；如果你需要重新启动服务器，之前你要查清现在的确没有人使用它，在发了邮件通知大家后，你才会去重启它，以免对他人的工作造成干扰；偶然遇到一个不是你负责的模块的bug，你会把它告诉相关人员，而不是让它溜掉；在你解释了两遍之后，对方仍然表示没有明白，你不会露出不屑的表情，而是再解释一遍，等等。专业都是在细节中体现，专业精神体现在让

事情做得更好的点点滴滴上。

说一句大一点的话，见笑了，我希望我们国家能够有越来越多的专业工程师，这样软件业才有希望。

在微软这样的大企业里，员工都是很独立的，员工之间在工作上的联系，就好像是两个小企业的联系，很正式。只要是为了完成一项任务，没有什么别的考虑，目标就是一个：把事情做好。作为一名员工，他（她）的所作所为，一方面是在做工作，同时也是在展现一个个体，每个人都想表现的专业和优秀。当他有了这种想法，就会对自己加以控制，向着专业的目标奔去，就好像自己给自己加了发动机，这种情况的产生大概就是近朱者赤吧。

我很难描述这种氛围是什么，大概是一种柔和而催人进取的氛围。我觉得，在这样的氛围中工作是一种幸运。更希望有更多的中国公司能提供这样良好的氛围。

附录B　新人工作指南

经过一番努力，当你终于在一家公司里拥有了一小块工作区域的时候，我想说的是：祝贺你！虽然可能只是一张办公桌，一台电脑，但这是平台，只要你足够努力，你就能发挥出你的价值。

快速融入公司

当我们到了一个新环境，我们首先要做的是快速地融入到环境当中。如果对什么都不熟，工作起来会不大方便。对于测试工程师来说，因为是技术这条线上的，所以我们习惯地去关注软件项目，并马上就会投入到具体的项目中，这样没有错，而且做得很对。然而，我们同时也要关注其他一些方面，以让自己快速地融入到公司当中。

首先要了解一下公司的组织结构，以及主要负责人的名字。例如，公司的总经理是谁？有几位副总？他们各自负责的工作是什么？有几个部门？部门负责人是谁？部门之间的分工是怎样的？自己的leader是谁？自己的部门下有多少正在进行的项目以及项目的大概内容是什么？了解了这些，我们对公司的整体框架就有了一个整体的印象。如果公司人力资源部的同事没有对你说这些，你可以主动去要求他们做这方面的讲解。

其次是要快速地与自己小组的同事熟悉起来。不管公司的规模大与小，在公司内部与我们打交道最多的是小组内的同事，小组有点像公司这个大集体中的一个“小家”。所谓的团队合作，也大多是指与小组内同事的合作。要想快速融入到团队中，沟通是一个不变的法宝。在各种场合，尝试着多表达自己的观点，包括邮件里、会议上等。另外，一些私下的场合的沟通也是很有用的，例如，中午一起吃饭时的沟通，或者遇到问题后而寻

求同事的帮助等。这些沟通能让我们的名字在大家的眼前“闪过”多次，从而达到快速彼此熟悉的目的。

再次是了解公司的各项规章制度。在一个公司工作，就要遵守它的规章制度。在入职的时候，我们可以主动去了解公司的一些制度，以方便将来遵守，例如考勤制度、请假制度、加班制度、报销制度、工作报告制度、考核制度，等等。

最后是快速了解公司周边环境。例如，公交车站、地铁站、银行、邮局、超市、餐馆、书店等等，知道这些信息会方便我们的生活和工作。

平稳度过试用期

现在基本上每个公司对新员工都会有一段时间的试用期，长短不一。经过N次面试得到了一个工作机会后，在试用期内千万不要松懈。大家可能都或多或少地听过军队里新兵受欺负的故事（例如新兵要给老兵洗衣服，或者帮老兵干活），或者是监狱里新囚犯受欺负的故事（例如挨打，上贡，倒马桶，睡在马桶边上之类的），这些故事都折射出一个现实，那就是融入新环境是不容易的。在公司的一个小组里，当然不会发生如同故事里那样的事情，但是，实际的环境对新人还是有些不利的，这种不利是人际关系上的。同样的一件事情，如果你和对方熟悉，他更能理解你，会往好处想，否则，更容易产生误解。例如，甲是老员工，甲让乙做一件事情，乙在规定的时间内没有完成，如果乙是老员工，甲可能会认为这件事情比较难，而如果乙是新员工，甲可能就会在心里想：是不是乙不会？如果这样的事情发生了几次，势必会影响甲对乙的评价。

在自己是新员工的情况下，要有“新兵”意识，多做一些杂事，保持积极而谦虚的状态，尽快博得小组的认同。这不是让大家弄虚作假，而是让大家避免让自己处于不利的情形。

有一点危机感

比尔盖茨说，微软离倒闭永远只有18个月。他这是在提醒微软的员工，虽然微软很有钱，但是要有危机感。当我们在公司工作的时候，也需要有一点危机感。

在工作中，每个人都会有抱怨，这很正常，但抱怨不能影响到实际的工作，该干什么还是要干什么，而且还要干好。公司觉得你“行”或者“不行”的结论从来不是以几天的印象来总结，公司对一位员工的评价是来自于员工中长期的实际工作的表现，例如3个月、半年，等等。对现在的人才市场来说还是属于买方市场，公司选择的余地比较大，所以，在实际的工作中带一点危机感，会减少或者避免我们最后被辞退的被动。

工作方法上的建议

下面是几条工作方法方面的建议，供你参考：

1. 保持积极的状态

不要总是等leader来给你下一些指令，我们要发挥自己的能动性。有一位leader曾经向我抱怨说，有的人只能完成她下达的任务，完成了以后就是“原地待命”，总是要等她来push（推动），她对这种状态很不满意。在工作任务做了分配以后，实际上我们就是第一责任人，也是最熟悉这一块工作的人，我们要保持积极的状态，除了完成指定的任务，还要看看还有什么要做的。只要是对项目和小组有益的事情，我们都要积极主动地去做。

2. 对领导保持透明

及时主动地向leader通报自己的工作状态，例如进展、遇到的难题、计划，等等。Leader只有在知道了每一个人的工作进度以后，才方便做小组层面上的安排，调配资源。如果你早于计划完成了任务，你就可以被安排去

支援同事或者做下一个任务的准备工作；如果你遇到了难题，卡在了一个地方，leader会来帮助你，或者让你暂时跳过这块“硬骨头”。但如果你到了最后“摊牌”的时间才向leader承认，自己没有完成任务，这样做是很危险的。因为这样不但你没有完成任务，你还会拖了整个小组的后腿。

3. 调动资源

要完成一个工作任务，有的时候有些事情会超出我们的能力范围，例如在硬件资源上或者技术储备上。当遇到了问题，我们就要积极地寻求帮助，调动资源。例如，如果遇到了一个技术难题，我们就可以把问题通过邮件发给本小组或者其他小组，然后一边自己钻研一边等待回复。再如，Leader就是一个很好的资源，他能调动的资源肯定比我们“当兵的”要多得多，而且他也会很在意你发出的帮助信号，所以遇到问题的时候不要不好意思，可以去寻求他的帮助。

4. 保持良好的协作

在小组内保持良好的人际关系，避免直接的冲突，这对于团队协作意义重大。

5. 不停地改进自己的工作

不要让自己处于停滞状态，当项目松弛下来或者有了空闲的时间，我们应当回过头审视自己的工作，总能找到可以改进的地方。

如何招聘测试工程师

可能有的读者朋友心想，我才刚步入这个行业，怎么会让我去招聘测试工程师？就想把这篇短文略过，这当然有道理。但是，即使是你刚入这个行业，去招聘测试工程师的机会还是有的。例如，如果公司只有你一个测试工程师，哪怕你刚到公司一个月，你可能也会参与招聘后续的人员的工作。再如，Leader临时有事无法面试，让你帮忙把把关，等等。所以，我

们还是要对如何招聘测试工程师做一个了解。

可能又有读者朋友说了，招人嘛，当然招高手了，什么项目都难不倒他的那种。真的是要招高手嘛？我们再来仔细想一想。高手诚然好，谁都想要，但是也有风险在里面：一是如果公司规模不大，待遇普通，不一定能达到高手的薪金期望；二是你的公司和项目不一定能够吸引住高手，如果一个高手工作几个月就走了，对公司来说一点价值都没有创造。所以，招人的时候不是要高手，不是要高学历，合适就好了。

最近一年来因为工作关系，我主持了很多次对测试工程师的招聘，对此有一些体会，我推荐的几条聘用原则是：

（1）候选人的性格和处事方式要与公司的文化相吻合。只有大致吻合，候选人才会对公司有认同感。公司各有不同，候选人也各有不同，只有认同相同规则的人才能合作得很好，所以我们要把容易抱成团的人选进来。

（2）具备完成项目的技术能力，或者基础好，有潜质，可培养。

（3）能够快速地融入到团队中并稳定地工作，这种沟通能力和团队协作能力甚至比技术本身更重要。

（4）把工作机会给珍惜机会的人。不管是哪一级的管理者都不可能天天盯着员工，日常工作只能靠员工自觉去完成，而珍惜机会的人会很主动，省去了不必要的说服和监督。如果你不想以后变成一个保姆，希望团队有战斗力，那请把工作机会给珍惜机会的人。

招聘测试工程师的时候，应当准备笔试。笔试的好处是可以检测出候选人的真实技术水平，而且对所有候选人都是一个尺度，很公平。另外，试卷是可以长期保存的，可供做最后决定的时候翻阅。面试的印象是不会留存多久的，只要候选人一多或者隔了几天，脑子里的印象就有可能变成一罐浆糊。

做好面试。在笔试的基础上，借助面试的灵活性，观察候选人的气质，考察候选人的表述能力、沟通能力、理解能力和协作能力，揣摩他（她）

今后在工作中的表现，是不是合适的测试工程师人选。

不要忘记了，在你面试候选人的同时，其实也是候选人在面试你。他们也在默默地观察你，通过你的表现去揣摩公司的氛围、技术实力和状况，只不过他们是被面试者，所以显得弱势一些、隐蔽一些。所以，如果你一看有了面试别人的机会，就大大咧咧、趾高气扬，你就等着别人笑话你吧。另外，你这样也很难招到人才，躲你还来不及呢，更别谈还要和你一块共事。

在文章的最后，我要重复我的一个观点：把工作机会给珍惜机会的人。

公司会如何选择测试Leader

前面说的都是具体工作方面的事情，现在说一说发展。我相信很多人都希望自己在职业道路上能有发展，对于测试工程师来说，离我们最近的一个职位是测试组的Leader。

既然拿出一部分钱来建测试组，就说明公司管理层不可能天天来盯着测试这一块，这个Leader就是公司在测试方面的实际领导。测试工作能开展到什么程度，和Leader有很大的关系。一个测试组Leader需要具有什么品质呢？

（1）对公司有认同感。对公司没有认同感的人，不会在考虑范围之内。

（2）应当具有测试经验。只有有了测试经验，他才有可能从测试实践中去提炼出管理思路。如果公司小，测试Leader不可能全职做管理，肯定还要做测试，而且是测试主力，没有测试经验怕是做不下来的。另外从实际情况来看，对测试经验的要求也不会太高，能胜任本公司的产品测试和测试管理工作就可以。人才，合适的最好。

（3）应当具有开发经验。从本质上讲，测试和开发是相通的，良好的测试能力来自于对软件产品方方面面的了解，尤其是产品的实现方式和内

部逻辑，这些正是开发人员所实现的。自动化测试是测试发展的方向，自动化测试实际就是测试方面的编码的代名词。

（4）应当有领导和管理意识。测试小组中需要测试高手，但是没有领导和管理意识的测试高手不适合做Leader。只顾着一个人跑，把大家都落下了，这样带不出一个团队来。

一个技术人员刚刚转入管理角色的时候，往往不自觉地就会带着纯技术的眼光去看问题，但Leader毕竟是一个管理职位，在管理方面还要做些学习，转换自己的思维，拓展自己的眼界。

请从现在开始为自己成为Leader做准备吧！

祝大家工作愉快！

附录C　简历模版

个人简历模版

简历—姓名

求职意向：软件测试工程师

姓名		性 别		出生年月	
民族		毕业院校			
学历		专业			
联系方式		邮件			

培 训			
时间	培训机构	培训内容	证书

技能特长		
技能	使用时间	技能水平

工作经历				
时间	公司名称	部门	职 位	职责描述

项目经历			
时间	项目名称	环境简述	角色描述

（备注：技能特长、工作经历、项目经历是重点部分，请尽量详细描述）

自我评价

自 荐 信

（以下可写一封简短的自荐信）

（备注：笔者编制此模板时参考了北京中讯汉扬公司的简历模板）

个人简历模版举例

简历——赵建国

求职意向：软件测试工程师

姓名	赵建国	性别	男	出生年月	85.8
民族	汉	现住址	北京海淀区西三旗		
毕业院校	xx大学	毕业时间	2007年6月		
学历	本科	专业	计算机科学		
联系方式	13201234567	邮件	zhaojianguo@163.com		

培训			
时间	培训机构	培训内容	证书
05/07-05/07	国家博物馆	如何成为一名优秀的义务讲解员	/
06/04-06/06	xx培训学校	MCSE	MCSE
07/03-07/04	xx外语培训学校	英语口语	口语培训结业证书

技能特长		
技能	使用时间	技能水平
软件测试	1年	熟悉
软件工程	1年	熟悉
C/C++	1年	熟悉
C#	6个月	掌握
SQL Server	1年	熟悉
Oracle	6个月	掌握

工作经历				
时间	公司名称	部门	职 位	职责描述
06/07-06/09	xx大学技术服务公司	研发部	测试工程师	在老师和老员工的指导下，全程参与图书馆管理系统的测试工作。并且还承担了用户手册的编写工作。
07/04-07/06	xx大学计算机系毕业设计第十小组	/	设计/开发/测试	与两个同学一起组成了毕业设计小组，我担任小组组长。我们一起完成了一个B/S结构的旧书交换系统。

项目经历			
时间	项目名称	环境简述	角色描述
06/07-06/09	图书馆管理系统	C/S结构，客户端编码以C#（2003）为主，后台数据库是SQL Server2000。	在这个项目中我担任测试工程师。我积极主动地学习，把课堂上学到的测试知识运用到实践中。我编写了测试计划和测试用例，并执行测试用例。因为是全程参与，所以对软件工程的实际运用有了一个了解，从整体上熟悉了软件测试的流程。
07/04-07/06	旧书交换系统（网站）	B/S结构，用ASP.Net来实现，后台数据库是Oracle9，实际运行时基于IIS。 这个项目同时也是学校团委实际定制的一个项目，自交付运行以来反馈良好。	这名项目是我们的毕业设计项目。小组共3人，我任组长。为了让大家都得到充分的锻炼，我们一同充当系统分析、设计、编码、测试和项目管理的角色，轮流担任。 在测试方面，我参与编写了测试计划和测试用例，并多次执行测试用例。

自我评价
✓ 计算机专业基础知识扎实，能满足测试工作对知识面广的要求；
✓ 有半年多的工作经验，了解软件工程的实际运用方法和软件测试的整个流程；
✓ 全程参与过软件测试项目，编写了测试计划和测试用例，多次执行测试用例，对软件测试工作有了更深的体会；
✓ 热爱测试工作，希望能在这个方向上得到发展；
✓ 积极，热情，好学；
✓ 勤奋，能吃苦；
✓ 注重团队协作，好相处；
✓ 英语读写能力良好，并具备一定的口语能力，参加过专业培训机构的口语培训。

自 荐 信

尊敬的xx公司：

我从网上看到贵公司在招聘测试工程师，特来应聘这个职位。我虽然是应届毕业大学生，但因为自己的积极主动，已经有了半年多的工作经验，掌握多项软件开发和测试技术。我参与过两个实际的软件项目，对软件工程的实际运用有了一定的了解，并熟悉了软件测试的整个流程。我编写过软件测试计划和测试用例，对软件测试工作有了深入的体会。在大学学习期间，我积极向上，勤奋好学，乐于助人，多次获得奖学金和助学金。

我非常希望能得到贵公司的面试机会。如果能有幸加入到贵公司优秀的团队中，我相信自己一定会出色地完成工作任务。

谢谢！